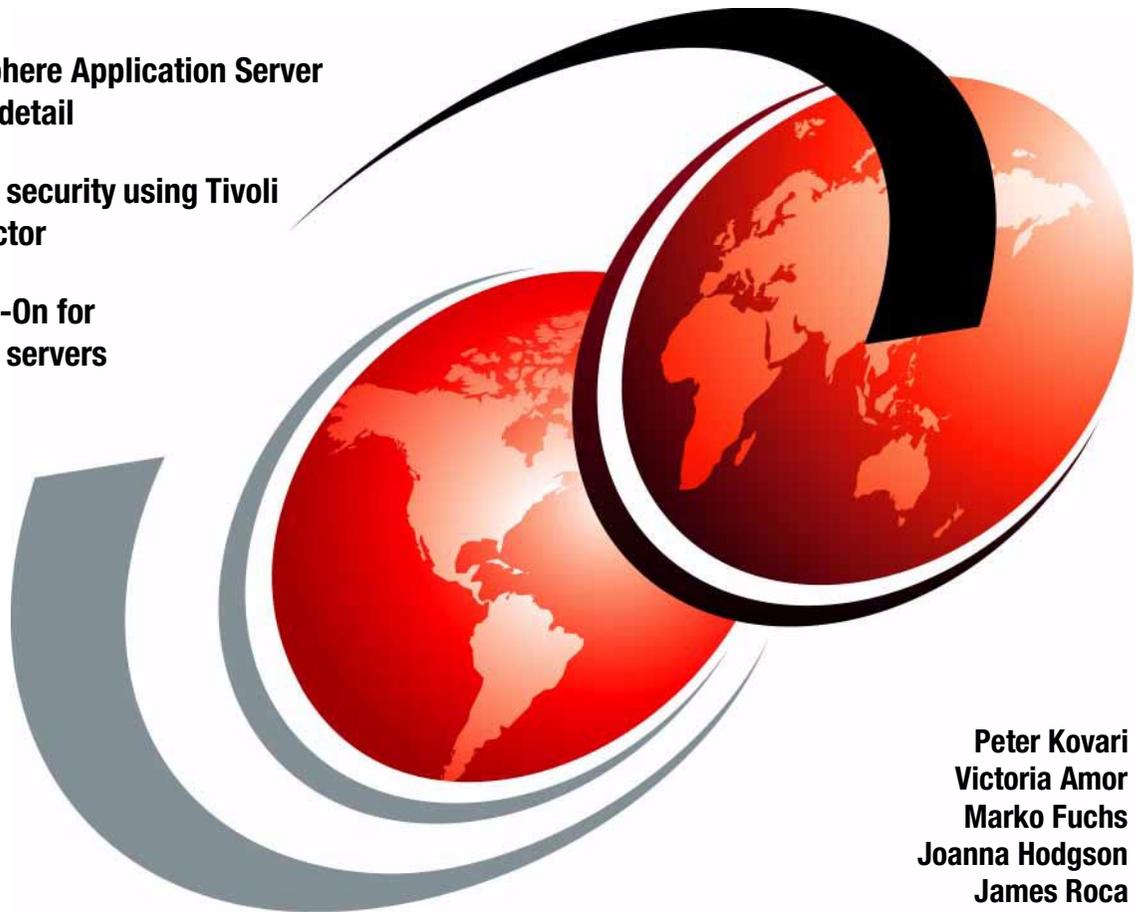


IBM WebSphere V4.0 Advanced Edition Security

IBM WebSphere Application Server
security in detail

End-to-end security using Tivoli
Policy Director

Single Sign-On for
application servers



Peter Kovari
Victoria Amor
Marko Fuchs
Joanna Hodgson
James Roca



International Technical Support Organization

IBM WebSphere V4.0 Advanced Edition Security

March 2002

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 521.

First Edition (March 2002)

This edition applies to WebSphere Application Server Version 4.0.1 Advanced Edition for use with Windows 2000, AIX 4.3.3, Sun Solaris 2.8; WebSphere Studio Application Developer V4.1 on Windows 2000; Tivoli SecureWay Policy Director 3.8 on Windows 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2002. All rights reserved.

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	xi
The team that wrote this redbook	xi
Notice	xiv
IBM trademarks	xiv
Comments welcome	xv
Part 1. Introduction	1
Chapter 1. Introduction to security	3
1.1 Security	4
1.1.1 Physical security	4
1.1.2 Logical security	5
1.1.3 Security policy	5
1.2 Security in use	6
Chapter 2. Security fundamentals	7
2.1 Authentication	8
2.2 Authorization	10
2.3 Delegation	11
Chapter 3. Security certificates	13
3.1 Public Key Infrastructure (PKI)	14
3.1.1 Encryption	14
3.1.2 Certificates	17
3.1.3 Elements of a certification authority system	19
3.1.4 Tivoli SecureWay PKI	20
3.1.5 Certification process	21
3.1.6 Infrastructure	23
3.1.7 Policies	25
3.2 Smart cards	25
3.2.1 Using smart cards	27
3.3 Where to find more information	28
Part 2. WebSphere security	31
Chapter 4. IBM WebSphere Application Server security	33
4.1 WebSphere security model	34
4.1.1 Security architecture	34
4.2 IBM WebSphere Application Server security features	39

4.2.1	How to secure an application	39
4.2.2	WebSphere authentication model	46
4.2.3	User registry	50
4.2.4	Security Center	52
4.2.5	Web Trust Association	62
4.2.6	Securing only the Administrative Server	63
4.3	WebSphere security and the operating environment	66
4.4	Performance considerations	67
4.5	Other security features of WebSphere	68
4.5.1	Encoded passwords	68
4.5.2	Security interoperability with z/OS	70
4.6	WebSphere Advanced Edition V4 ptf2	70
Chapter 5. The sample used in this book		73
5.1	Sample application: Webbank.	74
5.1.1	Base Webbank application structure.	75
5.2	Importing the sample into WebSphere Studio Application Developer.	77
5.3	Defining security roles.	81
5.3.1	Setting up users and groups in LDAP	82
5.3.2	Security roles with Application Assembly Tool	83
5.3.3	Security roles with WebSphere Studio Application Developer	86
5.4	Installing the Webbank application	89
5.4.1	Creating the application server	90
5.4.2	Setting up the data source	90
5.4.3	Installing the enterprise application.	91
5.4.4	Starting the application	93
5.5	Security role mapping	93
5.5.1	Security role mapping with the Security Center	93
5.5.2	Security role mapping during installation	94
5.5.3	Security role mapping with the Application Assembly Tool.	95
5.5.4	Security mapping with WebSphere Studio Application Developer	96
5.6	Domino Webbank sample	98
5.7	Security samples.	103
Chapter 6. Securing Web components		105
6.1	Static components served by a Web server	106
6.1.1	How to secure HTTP basic authentication for IBM HTTP Server	106
6.1.2	Managing access to IBM HTTP Server using .htaccess.	111
6.2	WebSphere Web module security.	112
6.3	Securing the Web components	117
6.3.1	Static pages served by WebSphere Application Server	117
6.3.2	Servlets	120
6.3.3	JavaServer Pages (JSPs)	124

6.4	Defining WebSphere Studio Application Developer security constraints	124
6.5	Configuring Web module security using WebSphere Studio Application Developer	127
6.6	Form-based and Custom Login facilities	131
6.6.1	Form-based login	131
6.6.2	Custom login	132
Chapter 7.	Securing EJBs	135
7.1	Securing EJBs	136
7.2	Assigning methods to roles	137
7.2.1	Configuring method permissions using AAT	140
7.2.2	Configuring method permission with WebSphere Studio ApplicationDeveloper	143
7.3	Setting up security role references	145
7.4	Configuring the delegation policy	147
7.4.1	Setting up delegation policy (Run-As mode) using AAT	149
7.4.2	Setting up delegation policy (Run-As mode) using WebSphere Studio Application Developer	150
7.4.3	Run-As mapping using the Administrator's Console	151
7.4.4	Run-As mapping during deployment	152
7.5	Topology considerations	153
Chapter 8.	Securing J2EE clients	155
8.1	J2EE clients	156
8.2	The Secure Association Service (SAS)	156
8.2.1	SAS on the client side	158
8.2.2	SAS on the server-side	160
8.3	Programmatic login	161
8.3.1	Client-side login	161
8.3.2	Server-side login	163
8.4	J2EE application client	165
8.4.1	Webbank J2EE client	165
8.5	Java thin application clients	166
8.5.1	Running the WebbankThinClient sample	167
8.6	Applet clients	168
8.7	Authentication summary	168
Chapter 9.	Securing Web services	171
9.1	Web services	172
9.2	Securing WebSphere Web services	172
9.2.1	Securing SOAP services	174
9.3	SOAP signature components	175
9.3.1	Web module	175
9.3.2	Envelope Editor	177

9.3.3	Signature Header Handler	178
9.3.4	Verification Header Handler	180
9.4	How to create secure Web services with WebSphere Studio Application Developer	183
9.4.1	Modifying the Webbank code	183
9.4.2	Creating the secure Web service	184
9.4.3	Testing the Web service	190
9.4.4	Generated code	191
9.4.5	The XML-SOAP Admin tool	192
9.4.6	Running the Webbank Web services sample	193
9.4.7	Checking the log file	194
9.5	Customizing the certificates for secure Web services	197
9.5.1	Certificates provided by WebSphere Studio Application Developer	198
9.6	Secure Web services samples in WebSphere V4 AE	199
Chapter 10. Programmatic security		201
10.1	Programmatic security	202
10.2	J2EE API	203
10.2.1	EJB security methods	203
10.2.2	Servlet security methods	204
10.3	CustomRegistry SPI	206
10.4	Trust Association Interceptor SPI	213
Chapter 11. Administering WebSphere Security		215
11.1	WebSphere Global Security	216
11.1.1	The Demo Keyring	217
11.1.2	Option 1: self-signed certificate using the IBM ikeyman utility	219
11.1.3	Option 2: certificate signed by a third-party CA	224
11.1.4	Configuring WebSphere to use your own keyring	229
11.1.5	Modifying the sas.client.props file	234
11.1.6	Enabling Global Security and securing the Administrative Server	235
11.2	Configuring the Web Server to support HTTPS	239
11.2.1	Generating a certificate to protect your Web Server	240
11.2.2	Configuring the IBM HTTP Server for SSL/HTTPS support	244
11.2.3	IBM HTTP Server (IHS) Cipher Support Strength	248
11.3	Client-Side Certificates for Authentication	253
11.3.1	Securing a Web Application to use client certificates	254
11.3.2	Obtaining a personal certificate	255
11.3.3	LDAP advanced security settings	258
11.4	Configuring SSL between Web server and WebSphere Application Server	270
11.4.1	Generating a self-signed certificate for the Web server plug-in	271
11.4.2	Generating a self-signed certificate for a Web Container	272

11.4.3	Exchanging public certificates	274
11.4.4	Modifying the Web server plug-in configuration file	275
11.4.5	Modifying the Web Container to support SSL	277
11.5	Restricting access to only HTTPS connections	280
11.6	Securing WebSphere LTPA with SSL	282
11.6.1	IBM SecureWay Directory Server	282
11.6.2	Creating a self-signed certificate for the SecureWay LDAP peer	284
11.6.3	Creating a key database for the WebSphere LDAP SSL peer	286
11.6.4	Modifying the SecureWay LDAP Directory to use SSL	287
11.6.5	Modifying WebSphere to use LDAP over SSL	290
11.6.6	Disabling SecureWay Anonymous LDAP searches	294
11.6.7	SSL and the Netscape iPlanet Alliance Directory Server	300
11.6.8	SSL Certificate creation with iPlanet Directory Server	301
11.6.9	Modifying iPlanet to support SSL/LDAPS	306
11.6.10	The iPlanet CA-signed certificate	308
11.6.11	Modifying WebSphere to support LDAPS with iPlanet	310
11.6.12	Disabling iPlanet Anonymous LDAP searches	314
11.6.13	SSL and Lotus Domino LDAP	317
Part 3.	End-to-end security solutions	331
	Chapter 12. Security in Patterns for e-business	333
12.1	Patterns for e-business	334
12.2	Access Integration pattern	335
12.2.1	Application patterns	336
12.3	Runtime patterns	338
12.3.1	Basic Runtime pattern	341
12.3.2	Runtime pattern variation	341
12.3.3	Single Sign-On Runtime patterns	342
12.4	Product mappings	346
12.4.1	Single Sign-On	346
12.4.2	Centralized security	349
12.5	More information	350
	Chapter 13. Policy Director	353
13.1	End-to-end security solutions	354
13.2	Using Tivoli Policy Director	355
13.2.1	Using Tivoli WebSEAL	355
13.2.2	Using Tivoli Policy Director to protect static pages	358
13.2.3	Using Tivoli Policy Director to protect WebSphere URIs	365
13.2.4	Policy Director LTPA authentication	368
13.2.5	Web Trust Association	381
	Chapter 14. Single Sign-On	393

14.1	Single Sign-On	394
14.2	WebSphere-Domino using SecureWay Directory	395
14.2.1	Enabling Single Sign-On for WebSphere	397
14.2.2	Enabling Single Sign-On for Domino	400
14.2.3	Implementing the security to the Webbank.nsf database	409
14.2.4	Testing Single Sign-On between Domino and WebSphere	411
14.3	WebSphere-Domino using SecureWay LDAP with SSL	418
14.3.1	Enabling SSO to use SSL in WebSphere	419
14.3.2	Enabling SSO to use SSL in Domino	420
14.3.3	Testing SSO between Domino and WebSphere using SSL	422
14.4	WebSphere-Domino using Domino LDAP	423
14.4.1	Installing and configuring software products and examples	426
14.4.2	Enabling Single Sign-On for WebSphere Application Server	426
14.4.3	Enabling Single Sign-On for the Domino Server	427
14.4.4	Implementing security to the Webbank.nsf database	428
14.4.5	SSO WebSphere-Domino using Domino LDAP with SSL	430
14.4.6	Troubleshooting Single Sign-On	432
Part 4.	Appendixes	435
Chapter 15.	Problem determination	437
15.1	The IBM HTTP Web Server	438
15.1.1	First steps	438
15.1.2	Problem determination	439
15.1.3	Web Server trace example	439
15.2	The WebSphere Web Server plug-in	440
15.2.1	First steps	441
15.2.2	Problem determination	441
15.2.3	Web Server plug-in trace example	442
15.3	The IBM WebSphere Application Server	443
15.3.1	First steps	444
15.3.2	Problem determination	446
15.3.3	JVM trace arguments	448
15.3.4	The Secure Association Service (SAS)	449
15.3.5	Post-analysis using the Log Analyzer	450
15.3.6	IBM WebSphere Application Server trace example	451
Chapter 16.	IBM WebSphere Application Server and LDAP	459
16.1	SecureWay Directory Server	460
16.1.1	Installing and configuring the IBM SecureWay Directory	460
16.1.2	Populating data entries in the IBM SecureWay Directory	467
16.1.3	Configuring WebSphere to use the SecureWay Directory Server	478
16.2	Lotus Domino 5.0	482
16.2.1	Configuring the Domino Server to run the LDAP service	482

16.2.2 Configuring WebSphere to use the Domino Directory	484
16.3 Netscape Directory Server	490
16.3.1 Adding a new user	490
16.3.2 Configuring WebSphere to use the Netscape Directory Server . .	496
16.4 Microsoft Active Directory	499
16.4.1 Adding a new user	499
16.4.2 Configuring WebSphere to use the Active Directory Server	504
Chapter 17. Using OpenSSL.	509
17.1 Open Source Software	510
17.2 OpenSSL.	510
17.3 How to create certificates using OpenSSL	511
17.3.1 Creating your own CA.	511
17.3.2 Client certificate.	513
17.3.3 Using the certificates.	514
Appendix A. Additional material	515
Locating the Web material	515
Using the Web material	515
System requirements for downloading the Web material	516
How to use the Web material	516
Related publications	517
IBM Redbooks	517
Other resources	518
Referenced Web sites	518
How to get IBM Redbooks	519
IBM Redbooks collections.	519
Special notices	521
Abbreviations and acronyms	523
Index	525

Preface

This IBM Redbook provides IT Architects, IT Specialists, application designers, application developers, application deployers and consultants with information to design, develop and deploy secure e-business applications using WebSphere Application Server V4 Advanced Edition.

Part 1, “Introduction” on page 1 provides a high-level overview of IT security, and introduces the basic and most important security services for the e-business applications.

Part 2, “WebSphere security” on page 31 focuses on IBM WebSphere Application Server’s security features and services. It provides a detailed overview of how to administer WebSphere security and how to secure Web components, EJBs, Java clients and Web services.

Part 3, “End-to-end security solutions” on page 331 offers details about end-to-end security solutions and application design. You will find an introduction of Patterns for e-business, in which security is in focus. We will also address Tivoli Policy Director and how to use it together with WebSphere Application Server. Single Sign-On is discussed in detail between application servers such as WebSphere and Lotus Domino.

The Appendixes on page 435 provide additional information on security, such as problem determination, installation and configuration of WebSphere and different LDAP servers, and how to use OpenSSL to run your own CA.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Figure 0-1 The IBM Redbook team (left to right: Marko Fuchs, Joanna Hodgson, James Roca, Victoria Amor, Peter Kovari)

Peter Kovari is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively about all areas of WebSphere. His areas of expertise include e-business, e-commerce, security, Internet technologies and mobile computing. Before joining the ITSO, he worked as an IT Specialist for IBM in Hungary.

Victoria Amor is an IT Specialist in IBM WebSphere and Lotus Domino within IBM Global Services in Spain. Her areas of expertise include IBM WebSphere support, particularly the areas of security and administration, and design and consultancy in Lotus Domino. She has worked within IBM for four years, participating in remarkable e-business projects such as the Sydney Olympic Games as the responsible of all Lotus Domino Servers in the Games System Center. Currently, she is part of the IBM Center for e-business Innovation, located in Madrid. In Raleigh, she was in charge of the execution of inbound e-business projects for clients.

Marko Fuchs is an IT Specialist with Haitec AG, Munich (Germany), a business partner of IBM. His areas of expertise include design and development of secure distributed applications using Web technologies, Enterprise Beans, Java Cryptography Extensions (JCE) and databases. He is also an expert on the e-business platform WebSphere Commerce Suite. Currently, he teaches the

store programming and store design courses for WebSphere Commerce Suite V5.1 in Germany and Austria. He holds a Diplom Informatiker (FH) degree from the Fachhochschule Rosenheim (Germany) and a Bachelor of Computing Science degree from Staffordshire University (UK).

Joanna Hodgson is an EMEA-level technical specialist in IBM WebSphere, based in the United Kingdom. She has eight years of experience in software and is an expert on IBM WebSphere, particularly the areas of security, integration with MQSeries and messaging systems through JMS. Her other areas of expertise include Java and OS/2. She has published a number of papers on WebSphere and contributed to IBM Redbooks, including the *OS/2 Debugging Handbook*. Joanna has a Bachelor of Sciences in Computer Science from Glasgow University.

James Roca is an Advisory IT Specialist and a senior member of the IBM EMEA IGS WebSphere Support Organization. He is tasked with providing the official support escalation channel for WebSphere problems originating in Europe, both defect and non-defect. He also provides on-site WebSphere consultation and support, typically for banks and financial institutions across Europe. His other areas of expertise include IBM AIX, Sun Solaris, IP networks and VPN firewalls. He previously co-authored the *WebSphere V3.5 Handbook*.

A remote team member has also contributed to the redbook project:

Tibor Vermes is an IT Specialist in IBM Hungary. He has two years of experience in the e-business field. His areas of expertise include PKI, smart card development, electronic commerce, and object-oriented programming. He holds an Msc. degree from the Technical University of Budapest.

Thanks to the following people for their contributions to this project:

Nataraj Nagaratnam, WebSphere Family Security Architect

Thanks to the following people from the International Technical Support Organization, Raleigh Center

Cecilia Bardy
Gail Christensen
Mark Endrei
Michele Galic
John Ganci
Carla Sadtler
Linda Robinson
Margaret Ticknor
Jeanne Tucker

Thanks to the following IBM employees:

Jonathan Adams, SWG Technical Strategy - IT Consultant

Axel Buecker, ITSO Austin - Security expert

Jeff Crume, IT Security Architect

Sharon Dagan, IT Architect

Jim Davey, Software IT Architect - Security

Tom Hyland, Software Services

Isabelle Mauny, EMEA WebSphere Software Services

Keith B. Smith, WebSphere Performance/Security

Notice

This publication is intended to help IT Specialists, IT Architects, application developers, application deployers and consultants to design, develop, deploy and manage secure applications using WebSphere. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Application Server 4.0.1 Advanced Edition, Tivoli SecureWay Policy Director 3.8. See the PUBLICATIONS section of the IBM Programming Announcement for WebSphere Application Server 4.0.1 Advanced Edition, Tivoli SecureWay Policy Director 3.8 for more information about what publications are considered to be product documentation.

IBM trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)® 

IBM ®

AIX®

DB2®

DB2 Universal Database™

Domino™

Everyplace™

FAA®

IBM.COM™

Lotus®

MQSeries®

Redbooks™

Redbooks Logo 

Notes®

OS/2®

RACF®

SecureWay®

SP™

Tivoli®

WebSphere®

z/OS™

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:
ibm.com/redbooks
- ▶ Send your comments in an Internet note to:
redbook@us.ibm.com
- ▶ Mail your comments to the address on page ii.



Part 1

Introduction



Introduction to security

Implementing and managing a secure e-business environment is one of the most challenging tasks today. The security paradigm has changed rapidly and nowadays it has influence on every aspect of an IT solution.

More and more businesses are running on an IT infrastructure or are supported by IT solutions. Running a business in any environment means that it has to be protected and secured completely.

Enabling security means enabling e-business, because users, customers and companies all need to be sure that their data cannot be corrupted or misused.

Companies need to come up with their solutions quickly, but on the other hand they also have to ensure a highly secure environment for operation.

This book will show you how to use WebSphere Application Server Advanced Edition's security features so as to satisfy the required security needs in an e-business solution. It goes further and covers some of the end-to-end security solutions tightly related to the WebSphere Application Server.

1.1 Security

This chapter is an introduction to basic security and provides no details or WebSphere specific information.

As new business practices emerge, most enterprises are finding that their existing security infrastructure is not capable of meeting the rapidly changing and more rigorous demands of business over the Internet. The demands of network security have now gone far beyond simply managing user accounts and restricting access between internal and external networks. These demands now require a sophisticated system that allows fine-grained access control to resources, yet is manageable enough to be tailored to protect systems from many types of security threats.

Security is a fairly vast topic; everything involves security to some extent, in a certain format. There are two main areas which have to be discussed separately:

- ▶ Physical security
- ▶ Logical security

Systems have to be protected both from outsiders and insiders. Do not forget that not every intrusion or attack is intentional; misuse of a system or improper administration can also cause damage.

1.1.1 Physical security

Physical security means protection against physical actions. It involves every physical element around:

- ▶ The machine(s) where the application is running.
- ▶ The room where the machines are operating.
- ▶ The building where the machines are installed.
- ▶ The site where the company is located.

The listed elements have to be secured against intrusion and damage, whether intentional or not.

Physical security also includes the protection of communication channels:

- ▶ Ground lines
- ▶ Wireless connection

The communication network has to be protected against eavesdropping and damage to the connection (cutting the line).

The subject of physical security goes much further than the objective of this book allows. This short section is only intended as a reminder of the concept of logical security.

1.1.2 Logical security

Logical security is related to the IT solution: the IT architecture and applications, including the business processes.

Communication

Network communication must be protected not only on a physical but on a logical level as well. Most of the companies' networks are connected to public networks. Therefore, applications are accessible from the outside world. Network level security must prevent unauthorized access.

Application

Securing an application is done on different levels. Security is designed from the very beginning of the implementation, when the processes and flows are designed.

- ▶ Securing the resources

This implies protecting the resources on an application level and exercising the security features of the runtime platform (authentication and authorization).

- ▶ Implementing the business processes securely

The processes have to be designed in a way that no weakness in logic can be found.

1.1.3 Security policy

Security policies are guidelines for an organization; they can be part of a widely accepted standard (ISO) or implemented by a certain organization or company.

Policies can define processes for different areas in an organization. Security policies focus on security related processes, for example, how to request a new password, how to renew a password, and so on.

These guidelines are very important in implementing a robust security for the whole system organization-wide.

1.2 Security in use

Since security is a complex and diversified topic, it is important to keep it simple.

The following list will show the basic security areas. These areas have to be taken into account and their requirements must always be fulfilled.

- ▶ **Authentication / Identification** - Measures designed to protect against fraudulent transmission and imitative communications by establishing the validity of transmission, message, station or individual.
- ▶ **Access Control** - The prevention of improper use of a resource, including the use of a resource in an unauthorized manner.
- ▶ **Privacy / Confidentiality** - Assurance that information is not made available or disclosed to unauthorized individuals, entities, or processes.
- ▶ **Integrity** - The correctness of information, of the origin of the information, and of the functioning of the system that processes it.
- ▶ **Accountability / Non-repudiation** - Assurance that the actions of an entity may be traced uniquely to the entity. This ensures that there is information to prove ownership of the transaction.
- ▶ **Administration / Configuration** - Methods by which security policies are incorporated into the architecture and the functionality that the system architecture needs to support.
- ▶ **Assurance / Monitoring** - Confidence that an entity meets its security objectives; this is usually provided through an Intrusion Detection System.
- ▶ **Security Management** - Assurance that an entity meets its security management objectives, processes and procedures.

If you keep this list in mind during design and development, security will be well implemented.



Security fundamentals

This chapter will discuss three fundamental security services also supported by WebSphere Application Server:

- ▶ Authentication
- ▶ Authorization
- ▶ Delegation

2.1 Authentication

Authentication is the process of establishing whether a client is valid in a particular context. A client can be either an end user, a machine or an application.

The authentication process involves gathering some unique information from the client.

There are three major groups of secure authentication used to gather this unique information:

- ▶ Knowledge-based - user name and password, for example.
- ▶ Key-based - physical keys, encryption keys, key cards.
- ▶ Biometric - finger prints, voice patterns or DNA.

Other authentication mechanisms can combine these; an example is digital certificates, where key-based and knowledge-based authentication are exercised.

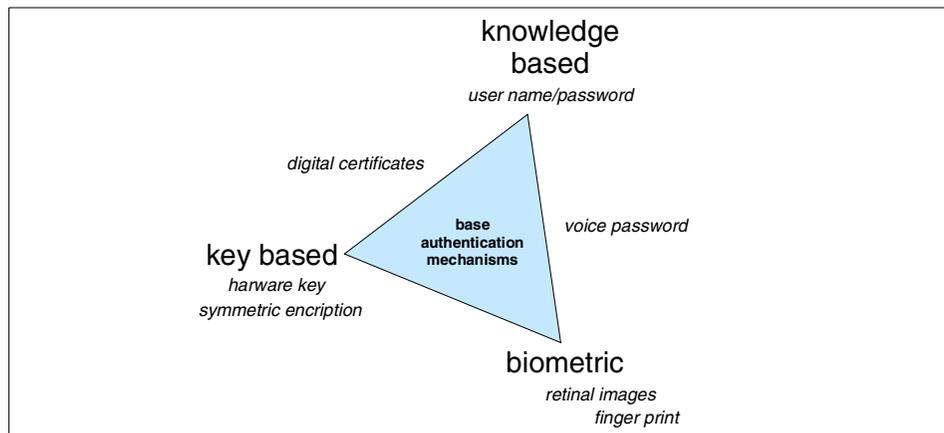


Figure 2-1 Base authentication mechanisms

The following paragraphs will discuss some of the authentication mechanisms used in IT systems.

User name and password

User name and password are the common method for authentication. The user who wants to access the system provides a user name and a password for login, which will be compared with the values stored in the system. The user name and password are presented as strings, which means that they can travel as plain text in the network. Sensitive information can be encoded, which is good for avoiding obvious security vulnerabilities. The use of SSL protocol for encrypting the user name and password represent an additional level of security.

Physical keys

Physical keys are objects that can be used to prove the identity of the object holder. Physical keys can be a piece of metal used to unlock your computer, a hardware device that is plugged into the computer to execute certain programs or smart cards that have a memory or microprocessor embedded.

Biometric authentication

Biometric authentication is the use of physiological or behavioral characteristics used to verify the identity of an individual. The biometric authentication consists of comparing the physical characteristics of an individual against the values of those characteristics stored in a system. The advantage of this type of authentication is that the user cannot lend its identification values, as can be done with a physical key, and cannot forget it like a password.

The disadvantages are that a biometric authentication system is expensive and most of the biometric methods such as voice print, finger prints, retinal images or facial images require that the user have a measurable characteristic.

Digital certificates

Digital certificates are based on public/private key technology. This technology uses a pair of keys that work together. What the public key encrypts, only the corresponding private key can decrypt and vice-versa. The public key is available for other users that can use this key to encrypt information to send to an individual. The private key is only accessible for one user, the key holder, and can be used to decrypt messages encrypted with the public key sent by another user or to encrypt messages that can only be decrypted with the corresponding public key by the other communicating party.

The digital certificates are used to automate this process and can be installed on the client or on the server machine. Certificates are usually issued by a Certification Authority (CA) that guarantees that the owner of the certificate is who he/she says he/she is.

2.2 Authorization

Authorization is the process of checking whether the authenticated user has access to the requested resource. There are two fundamental methods for authorization:

- ▶ Access Control List

Each resource has associated with it a list of users and what each can do with the resource (for example: use, read, write, execute, delete or create).

Usually, an Access Control List specifies a set of roles allowed to use a particular resource and also designates the people allowed to play these roles.

For example, in a bank account object, we can have different methods (transfer, deposit, *getBalance*, *setInterest*, etc.). The access right can be granted on the basis of the roles of the users within the organization. A bank teller can have access to the *getBalance* method but not to *setBalance*, while a manager can access to both methods.

Table 2-1 Example of a Role Access Control List

Resources	Bank teller role	Manager role
getBalance method	yes	yes
setBalance method	no	yes

- ▶ Capability list

Associated with each user is a list of resources and the corresponding privileges held for the user.

In this case, the holder is given the right to perform the operation on a particular resource.

In the previous example of the bank account object, the access right is granted to the user if the resource is listed in the user's capability list.

Table 2-2 Example of a capability list

Roles	getBalance method	setBalance method
Bank teller role	yes	no
Manager role	yes	yes

You will find the two tables shown above very similar, but the rows and the columns are switched. Actually, this is the difference between the two approaches. We have two sets: roles and resources. In the first case, roles are mapped to resources, while in the second case resources are mapped to roles.

The access control list is exercised generally, because managing security for certain resources is easier and more flexible than mapping resources to roles.

2.3 Delegation

Delegation is the ability to leave an intermediary to do the work initiated by a client according to a delegation policy.

For example, in a distributed object environment, a client can request the method of an object on Server A. The method request results in invoking another method of an object in server B. Server A performs the authentication of the identity of the client and passes the request to server B. Server B assumes that the client identity has been verified by server A and responds to that request as shown in Figure 2-2.

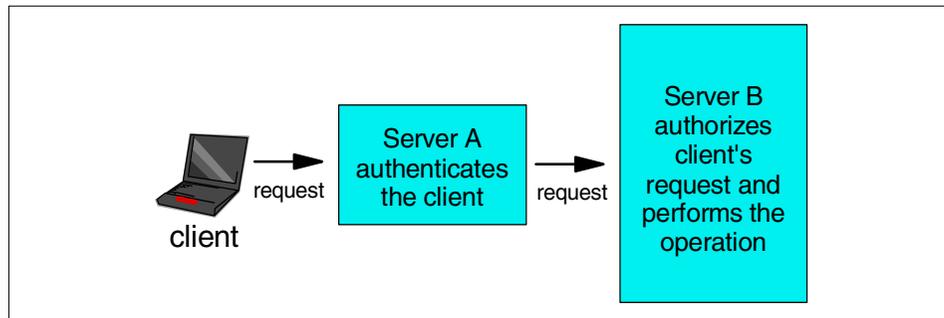


Figure 2-2 Delegation mechanism

Depending on the application environment, the intermediary can have one of the following identities when making a request to another server:

- ▶ Client identity: the identity under which the client is making the request to the intermediary.
- ▶ System identity: the identity of the intermediary server.
- ▶ Specified identity: identity specified through configuration.

Note: Delegation is not defined in the J2EE 1.2 specification.

The most common case for delegation between EJBs is the Facade design pattern. One implementation of the pattern is when a session EJB is provided to handle the processes using entity EJBs. The session EJB is simply working as an interface and hides the complexity of the entity EJBs (see Figure 2-3.)

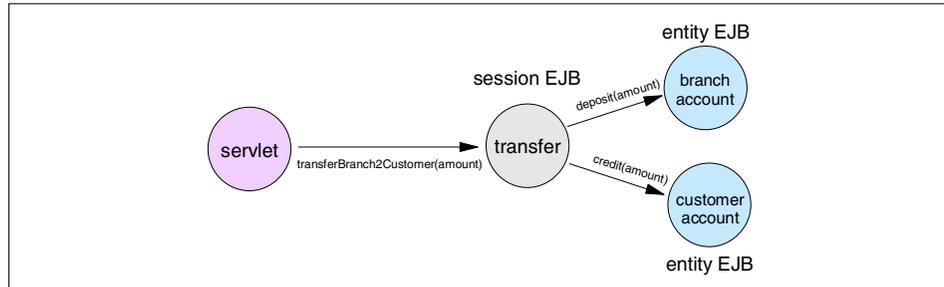


Figure 2-3 Facade pattern realization with EJBs

To see another example of delegation in WebSphere, see the WebSphere delegation model section in *IBM WebSphere V4.0 Advanced Edition Handbook SG24-6176*.



Security certificates

This chapter will cover the possibilities of increasing security in e-business systems.

The main motivation for developing more reliable security systems is the very fast growth of the Internet. A huge amount of information must be transmitted through the network. Internet security comes into play when the data we would like to hold or transmit is confidential; typically, this situation arises during banking or inter-enterprise business transactions, or in the course of electronic commerce.

IT security has many aspects and levels. The lowest level involves the encryption standards. These are based on very complex mathematical algorithms (for example, the famous RSA algorithms). The security protocols (such as SSL) are one level higher and also use the mentioned algorithms. These protocols are developed to transmit data within an unreliable environment. The complex security systems use many protocols and algorithms for user authentication and data encryption.

3.1 Public Key Infrastructure (PKI)

This chapter provides a brief overview of Public Key Infrastructure (PKI). PKI is a part of IT security and today's security needs bring it into focus.

3.1.1 Encryption

PKI is closely related to cryptography. Although it seems complicated, it is not. We do not need to use low-level mathematical algorithms, but we do need to understand the background involved.

During World War II, spies and army commanders used encrypted messages to transmit their messages. The receiving party used a pattern to decode the message. Nowadays, we do not use patterns to encrypt and decrypt messages, but the methodology is the same. We have the confidential information in plain text format and would like to hide this information from unauthorized users. We need to encrypt the text. The result of encryption will be the cipher text. This is information that is not readable without the encryption key. The receiving party uses another key to decrypt the message. The encrypting and decrypting functions are comprised of very difficult mathematical algorithms and the keys are represented by large numbers.

Secret key cryptography

The secret key algorithms were invented earlier than were the public key algorithms. They use one key to encrypt and decrypt the data.

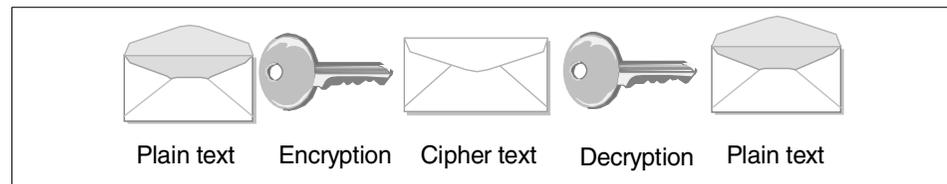


Figure 3-1 Symmetric key encryption

Figure 3-1 illustrates the concept of symmetric key cryptography. The algorithms used provide a great advantage: they are faster than the public key cryptography introduced later. They have a considerable disadvantage as well: the same key is needed for encryption and decryption, and both parties must have the same keys. In today's cryptography, the secret keys do not belong to persons but to communication sessions. At the beginning of a session, one of the parties creates a session key and delivers it to the other party; they can then communicate securely. At the end of the session, both parties delete the key and, if they want to communicate again, they must create another key.

The following section will discuss how to secure the delivery of the session key.

Public key cryptography

The first imperative of public key cryptography is the ability to deliver the session keys securely. It has many more benefits than private key cryptography, as we will see in the following sections.

Public key cryptography involves the use of different keys for encrypting and decrypting functions. If you encrypt something with key 1, you can only decrypt it with key 2, as shown in Figure 3-2.

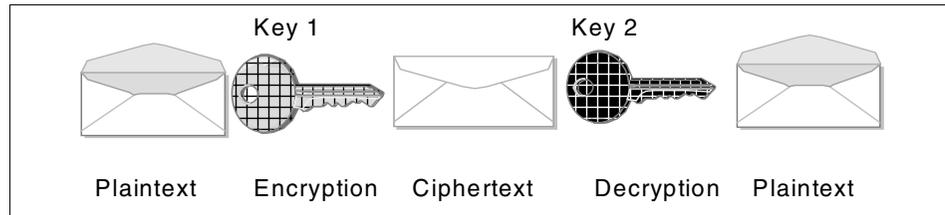


Figure 3-2 Public key concept

This architecture allows the use of one of the keys as a private key. This means that nobody can have access to this key except the owner. The other key can be used as a public key. If a user wants to send an encrypted message to another person, he or she will get the other person's public certificate, encrypt the message and send it. The message can be decrypted only by the owner of the private key.

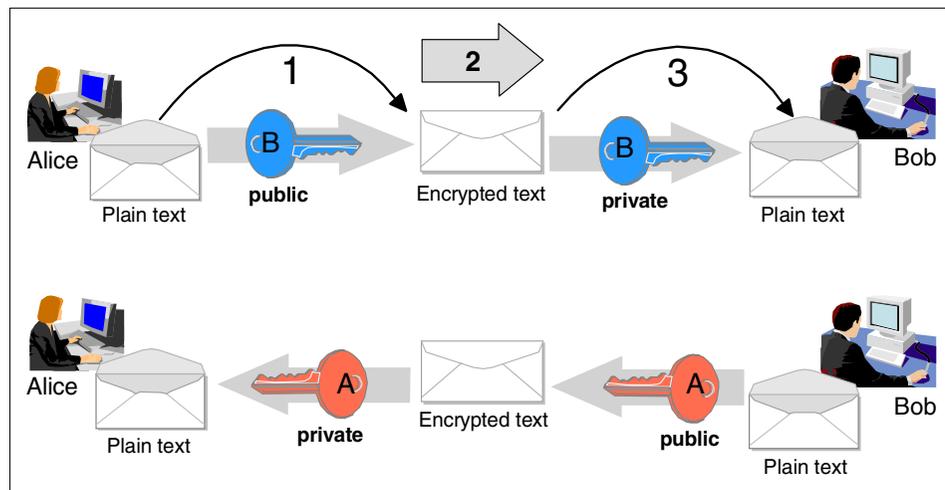


Figure 3-3 Using private key cryptography

Figure 3-3 shows a sample communication between two persons: Alice and Bob.

1. Alice wants to communicate with Bob but she does not want anybody to read the messages. She will use Bob's public key to encrypt the message.
2. Alice sends the message to Bob.
3. Bob uses his private key to decrypt the message.

If Bob wants to answer, he should use Alice's public key for encryption.

The example above is not suitable for the encryption of large amounts of data, because public key algorithms are very slow. We use the private key algorithms to transmit large amounts of data. The session keys must be delivered with the public key algorithm and will be used during the communication.

Figure 3-4 provides a very simplified overview of keys delivery using Secure Sockets Layer (SSL).

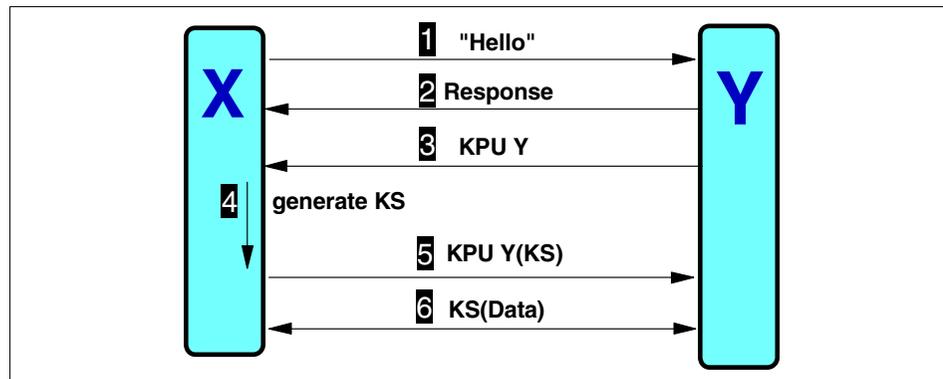


Figure 3-4 Delivering session keys using SSL

The process follows this pattern:

1. Communication is requested.
2. There is a response from the other party.
3. The partner (Y) sends the public key (KPU Y).
4. A session key (KS) is generated.
5. The session key (KS) is encrypted with Y's public key and sent (KPU Y(KS)).
6. The data transfer can start; data is encrypted with the session key (KS(data)).

This protocol solves most of the problems involved in setting up and running a secure communication. There are some other security problems which cannot be solved simply with encryption.

The most significant security issues are:

- ▶ **Authentication:** this means that we can identify the other party; we can be sure that they cannot pose as another person (or application).
- ▶ **Data integrity:** this means that no changes to the data are allowed during transmission, or that the changes should be recognizable.
- ▶ **Non-repudiation:** the sender cannot deny being the source of the information.
- ▶ **Data confidentiality:** only authorized persons can have access to the information.

Following is a practical example of the possible security breaches on the Internet:

The third step of the example mentioned in Figure 3-4 was that the sender sent his public key. If a user can intercept this key, then that user can act as the sender. He can remove the public key from the message and replace it with his own public key. He is then able to monitor all the data. This problem is called the “man in the middle” problem. It shows that it is not enough to encrypt messages; we must be sure of whom we are communicating with.

3.1.2 Certificates

A certificate is a document from a trusted party which proves the identity of a person. PKI certificates work in a similar fashion; if someone has a certificate from a trusted party, we can make sure of his or her identity.

Signatures

Signatures also work as in everyday life. Signatures used in the PKI environment work as follows: the information encrypted with a person’s (the sender) private key will be unique to this person. Anybody can decode the message, and the source will be identified, because only one public key can open the message: the sender’s public key. This message is almost good enough to be used for a digital signature; the only problem is that we would like to sign documents, and an encrypted document is too long to be a signature.

Hash (digest) algorithms

The solution for the problem of the long document is the hash functions. These algorithms are applied on plain text and the result will be a message digest. These mathematical algorithms have the following characteristics:

- ▶ The length of a message digest is independent of the length of the input text. We will get a fixed size output.
- ▶ The input of the function is not computable, knowing the output.

- ▶ One bit change in the input text will cause more than half of the output bits to be changed: small changes can be noticed easily.
- ▶ The most popular hash algorithms are MD5 and SHA1.

Using hash algorithms, the creation of signatures will follow this procedure:

1. The sender gets the plain text and digests it. The result is a so-called *message digest*.
2. The sender encrypts the message digest with his or her private key.
3. The sender sends the message and the signature to the receiving user.

The receiver should do the following at the verification phase:

1. Receive the message and the signature.
2. Decrypt the signature with the sender's public key and get the message *digest*.
3. Hash the message with the same method used by the sender and get the message digest (let us call it *digest2*).

If message *digest* and message *digest2* are identical, then the signature is valid.

Signatures are not enough for identification. For example, if someone wants to travel by air, a passport will have to be shown as proof of identification.

The certificate, similar to a passport, is issued by a trusted authority. It should contain information about the owner and should be signed by the authority.

There is a standard defining the form of a certificate, called X.509. This standard also defines the attributes of a certificate.

Some of the certificate extensions are:

- ▶ X.500 name
- ▶ Issuer X.500 name
- ▶ Distinguished name
- ▶ Issuer distinguished name
- ▶ Serial number
- ▶ Public key

From version 3 of the X.509 protocol, it is possible to include some specific extensions into a certificate. The most significant extensions are:

- ▶ **Key usage:** this signs the purpose of the certificate, for example:
 - Digital Signature
 - Non-repudiation

- Data encipherment
- Key encipherment
- ▶ **Basic constraints:** this relates to certification paths and designates the subordinate Certificate Authority (CA) certificate number to the End-Entity certificate.
- ▶ **Certificate classes:** this extension shows the class of the certificate (for example: class1,class2, etc.). The value indicates the trust level and the registration method of the certificate defined in the policy of the Certificate Authority (CA).

3.1.3 Elements of a certification authority system

A PKI system completes the tasks related to public key cryptography. These tasks should be separate, meaning that a PKI system should have some well-defined units to execute the different tasks. In some cases, the PKI implementation must separate the different functions physically (for example, in a commercial CA system). In this case, the elements listed below are located on different servers.

The logical elements of a PKI system are:

- ▶ Certificate Authority (CA)
- ▶ Registration Authority (RA)
- ▶ Certificate Repository (CR)

Certificate Authority (CA)

The CA component is the heart of a PKI system, it provides the “stamp” to the certificate. In some implementations, the CA component is issued together with the Registration Authority (RA) component. It stores its private key and can sign the certificate requests with it. This private key should be kept in a very secure place. If this key is corrupted, the whole certification tree will be unusable. It is possible to store this key on separate hardware.

Note: IBM has a cryptographic co-processor (IBM 4758PCI) for storing the private key and executing the cryptographic functions. This module is designed to meet the FIPS PUB 140-1 level4 specification. The card is able to keep the CA private key securely (any data tampering will be recognizable) and to execute encryption algorithms. Therefore, the private key does not need to leave the card and execution will be faster.

Registration Authority (RA)

This component is responsible for the registration process. It is an optional component of a PKI system but, in most cases, it is implemented. The main RA task is the verification of client requests.

Certificate Repository (CR)

This component is often called a *certificate directory*. The users of a PKI system use the issued certificates to authenticate themselves. If someone receives a signed message, the receiver will check the signature. If the signature was issued by a trusted party the message will be considered a trusted message. Otherwise, there is a problem. The certificate could have been revoked for certain reasons (the owner left the company, owner's private key was corrupted, etc.). In this case, the certificate should not be considered to be a trusted one. This problem is solved by publishing certificates in the certificate repository. When a user receives a message with a certificate, the validity of the certificate can be verified.

The list of revoked certificates is called Certificate Revocation List (CRL) and is usually stored in the Certificate Repository (CR). The most common way of implementing a CR is to use the Lightweight Directory Access Protocol (LDAP) standard (RFC2587).

Certificate Revocation List (CRL)

The CRL is a list in which the revoked certificates are stored. The widely accepted format of CRLs is PKIX X509 V2. The CRL should contain:

- ▶ The Certificate Revocation List (CRL) version ID
- ▶ The issuer's signature algorithm
- ▶ The issuer's name
- ▶ The update time and date
- ▶ The time and date of the next update
- ▶ Some extensions
- ▶ The signature of the issuer
- ▶ The serial number of revoked certificates

The time period for publishing the CRL is a parameter of the Certificate Authority (CA).

3.1.4 Tivoli SecureWay PKI

Tivoli SecureWay PKI is a fully functional PKI system. The previous version was called the IBM SecureWay TrustAuthority. One of the strengths of the system is its modularity.

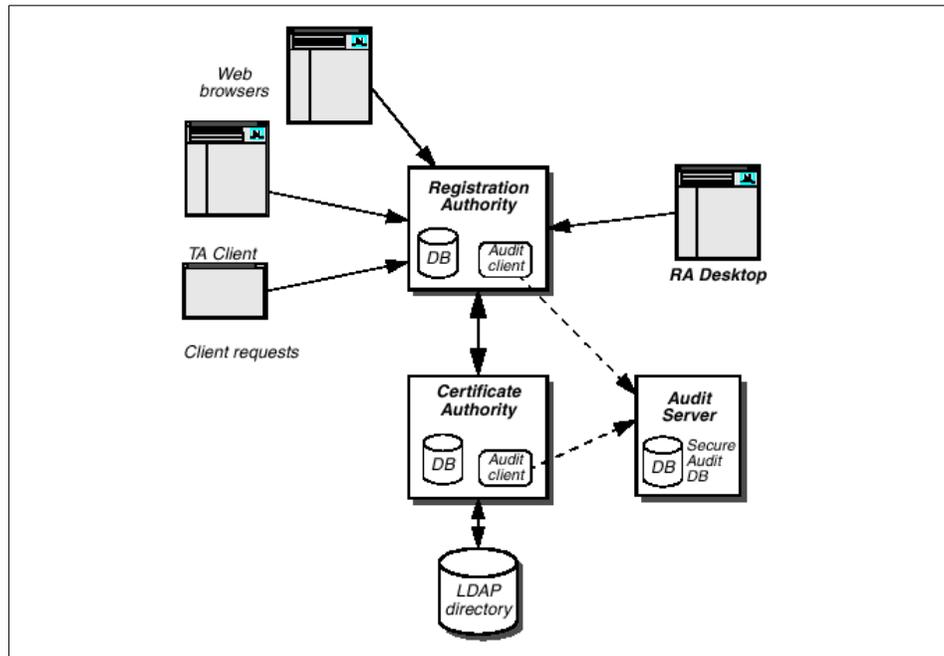


Figure 3-5 Tivoli SecureWay PKI

The structure of the system is shown in Figure 3-5. It is possible to separate the Certificate Authority (CA) and Registration Authority (RA) subsystems. The Trust Authority (TA) Client allows the users to manage their own certificates.

Tivoli PKI has a Java-based element: *RA Desktop*. It is the configuration interface for the system. RA administrators can manage the certificates using this component.

There is also a highly scalable LDAP directory included: Tivoli SecureWay Directory.

All events in the system are logged using the Audit Server component.

3.1.5 Certification process

Usually, there are two methods to issue certificates. The difference between the processes is the location where the client's private key will be generated.

In the first case, the client key pair is generated on the client side (on the client machine). The client will create a certificate request. The certificate request contains some information about the client (public key, name, e-mail address, key usage, some optional extensions, etc.). The request is signed with the

private key of the client and sent to the server. The server identifies the client before issuing the certificate. The first step is to verify whether or not the signature at the end of the request is valid (the public key in the request can be used for validation). If no error is encountered, then either the certificate can be issued or another client validation process can be started. The most secure method of client validation is for the client to appear personally and certify themselves at the authority location. If the client certification is successful, the certificate for the public key is created with the desired key usage. The client can download the certificate into his/her browser registry or onto a smart card. Figure 3-6 illustrates the data flow for the process.

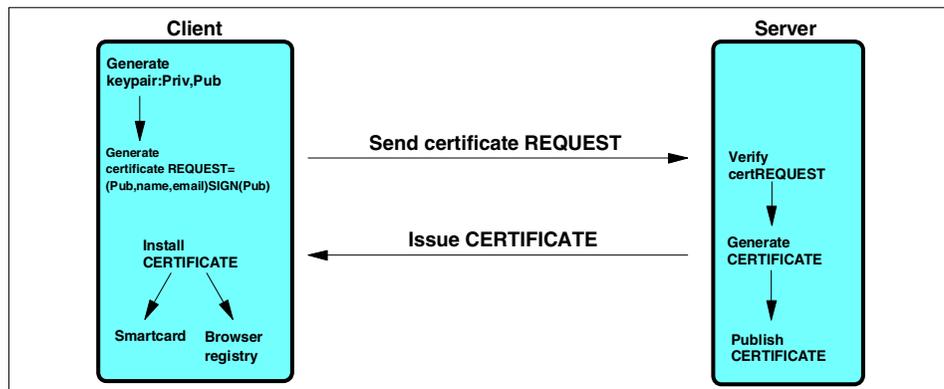


Figure 3-6 Issuing certificates

The other way to issue certificates is to execute the key generation process on the server side. This means that private keys should be created on the server side. This solution presents some problems:

- ▶ The key generation requires a lot of computing power. There should be very powerful computers applied as Certificate Authority (CA) machines or key generation will be very slow (in case of multiple requests).
- ▶ The private key must be issued and sent to the client, creating a weak point in the security.

There are situations when this method is better for issuing certificates. For example, let us imagine a research institute with a few hundred employees. The institute wants to make the entrance of the building more secure and also wants the computers to be used by the right persons. The company considers using smart cards for solving both problems. A PKI system can be implemented and every employee can get a smart card with a certificate and a private key. Obviously, the company will not establish a Web registration module for the employees (because of the fixed and small number of certificates to issue), but it will create the keys and certificates, install them on the cards and issue the cards

to the customers. This process does not have any weak points, because the cards will be given personally to each proper person. Smart cards usually do not allow the exporting of private keys, so they cannot be corrupted (unless the card is stolen).

3.1.6 Infrastructure

A Public Key Infrastructure (PKI) system acts as a trusted third party authentication system. It issues digital certificates for the communication parties (for users and applications). Some of its tasks are:

- ▶ Issuing of certificates
- ▶ Revoking of certificates
- ▶ Renewal of certificates
- ▶ Suspension and resumption of certificates
- ▶ Management of issued certificates
- ▶ Issuing a list of revoked certificates
- ▶ Protection of the private key

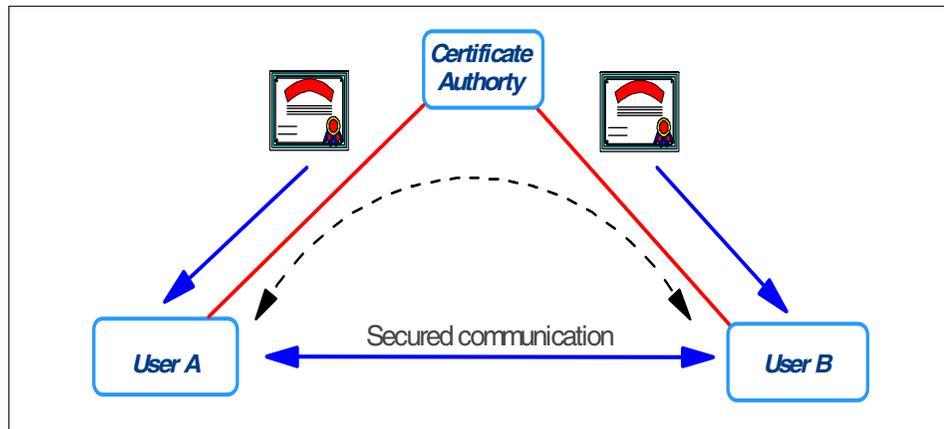


Figure 3-7 Simple CA architecture

Figure 3-7 shows the concept of the PKI, where both users belong to the same authority. The Certificate Authority (CA) issues certificates. The communication partners can use the issued certificates to certify themselves. The authority publishes the issued certificate in an accessible place.

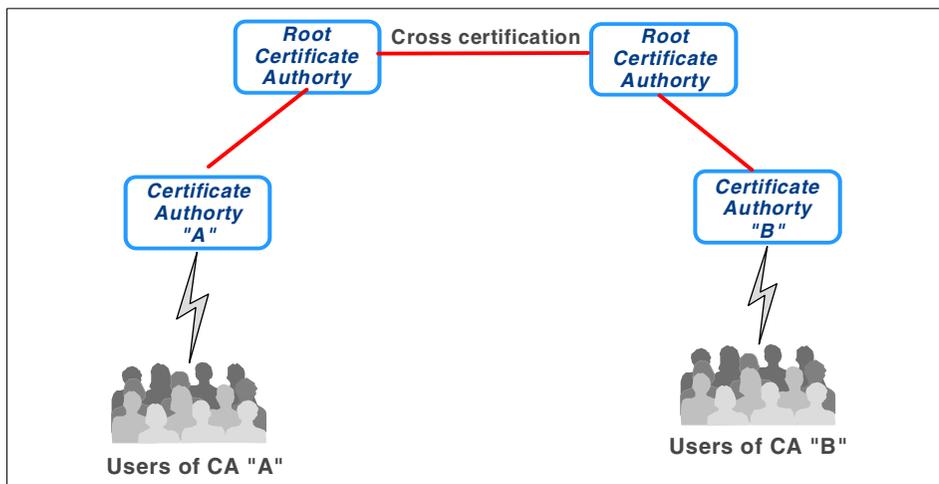


Figure 3-8 Cross certification

Figure 3-8 shows the concept of cross certification. Cross certification is useful when there is a need to build secure and authenticated communication between different organizations. For example, some banks can agree to accept transaction requests from users of other banks. In this situation, they must identify the users of the foreign bank. The CA of the bank can get the certificate of the user and will see that the certificate was issued by a trusted Certificate Authority (CA), so it can grant the access.

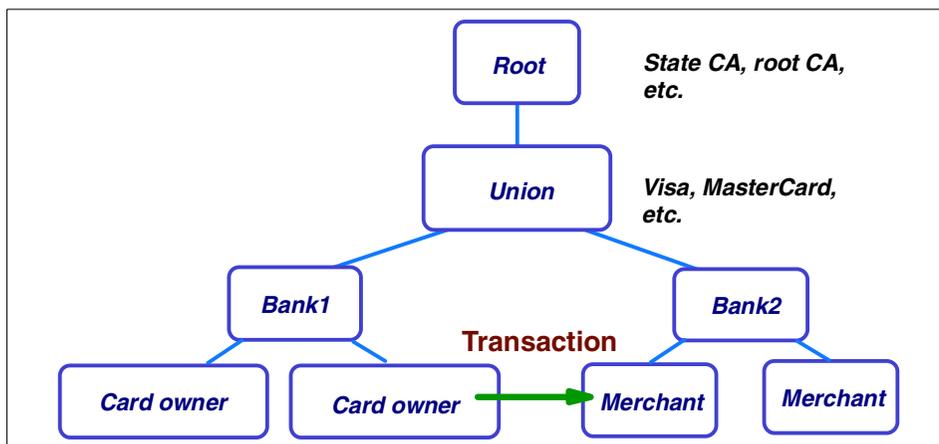


Figure 3-9 Hierarchical model

The other method of connecting Certificate Authority (CA) systems is to have a hierarchical architecture, as illustrated in Figure 3-9. For example, a bank can have several branches (in different countries), these branches belonging to different CAs. If the bank wants to introduce e-banking, it has to connect the different CAs. Let us assume that the user is abroad and wants to use e-banking. The user has to connect to the branch in the foreign country and perform the transaction. The CA will try to verify the certificate. Since the systems belong to the same root authority, and the certificate was issued by a trusted CA, it will be considered a trusted certificate.

3.1.7 Policies

No Public Key Infrastructure (PKI) system can work without a supporting policy. The policy describes the rules the system should follow. The policy should describe the physical environment of the system, the administration roles of the servers, the working processes, etc.

Usually, publicly used CAs publish a part of their policy called the Certificate Practice Statement (CPS). The CPS describes all functions, rights and processes related to certificates.

Verisign is one of the largest PKI providers. Verisign's CPS can be found at the following site: <http://www.verisign.com>.

3.2 Smart cards

Smart cards are the most commonly used enhancements of the PKI systems. They look like credit cards and are the same size.

What is the difference between credit cards and smart cards?

The widely used credit cards have a magnetic bar holding the information about the owner. There is a more solid version of such cards: memory cards. They hold the information in an EPROM chip. Smart cards hold the information and also have computing capabilities; they can execute certain functions.

What parts does a smart card have?

A smart card has to have capabilities for holding the data and executing some tasks relating to it. The file structure of a smart card is similar to that of the well known personal system (for example: DOS).

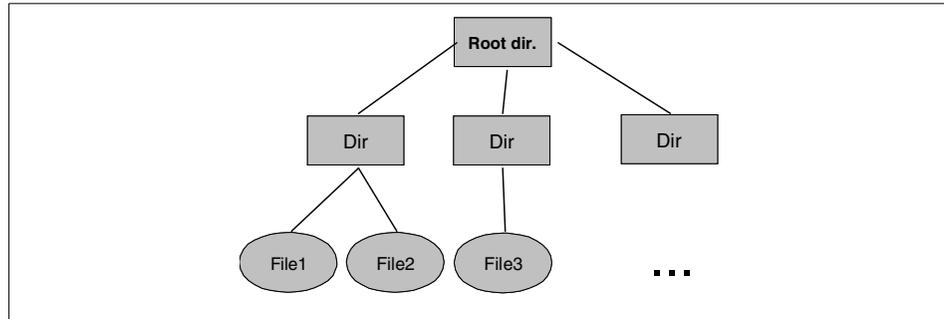


Figure 3-10 Smart card file structure

The *root* directory is often called the Master File (MF). There is only one MF per token (smart card). It can hold other directories, Dedicated Files (DF). The DFs can contain files called Elementary Files. Elementary Files can be public or secret keys, certificates, purse files (for banking purposes), etc. The most often used file type in a PKI environment is the public key file. Files can be protected against unauthorized access with PIN numbers; PIN numbers can be stored in Elementary Files called Secret Code.

A public key file consists of a private key, a public key and some certificates. Smart cards have two areas: one non protected and one protected. The non protected area is for holding public keys and certificates, the protected area is for holding private keys. There is a big difference between the behavior of the private keys and other files. All files can be imported to the card, but the private keys cannot leave the card. If you import a private key to the card and erase the original one, the key will disappear. You can use only the crypto functions of the card for encryption and signing.

What functions are available on the card?

The difference between memory cards and smart cards is related to the functions smart cards can execute. These functions are:

- ▶ Decrypting and encrypting mechanisms
- ▶ Signing mechanisms
- ▶ Key generation mechanisms

Smart cards usually have a device, called a smart card reader, which can be connected to computers. Most cards implement the PKCS11 standard interface for communication with the card.

Note: Smart card readers can be connected to communications ports (COM) or to USB ports. There are some types of cards which do not need a reader; these can connect directly to the USB port.

What does a communication look like between a program and a smart card?

Figure 3-11 illustrates the method of signing:

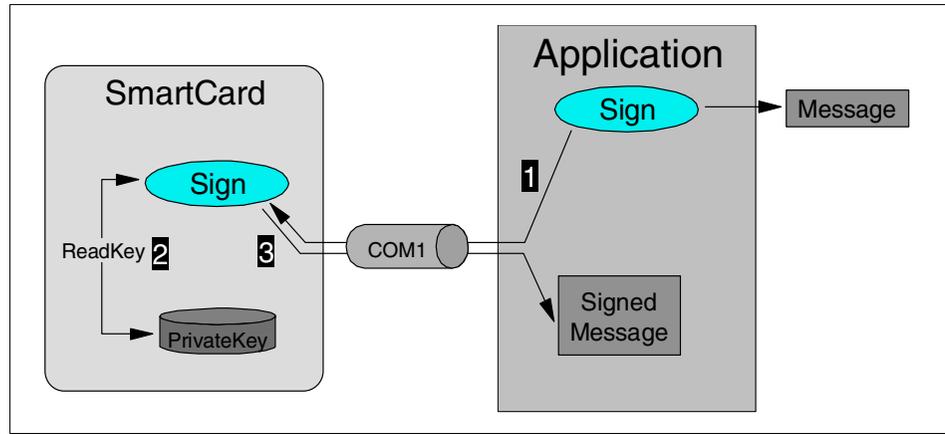


Figure 3-11 Signing with a smart card

The process flows as described below:

1. The application transmits the hashed data to the card.
2. The card computes the signature from the private key stored on the card.
3. The signed data gets back to the calling application.

3.2.1 Using smart cards

There are several different levels of security in a PKI system. If smart cards are not used, the generated private keys will be located for example in the Windows registry or in the Netscape registry. This is enough for a medium security application. If there is a need for establishing high-level security, this solution will present some disadvantages. The biggest disadvantage is that a computer can be used by another person, and he or she can get the private key easily.

After this, he or she can sign messages as the owner and can decrypt the owner's messages.

The situation is more secure if the private key is on a card and can never leave it, because the owner can more easily keep the card physically secure.

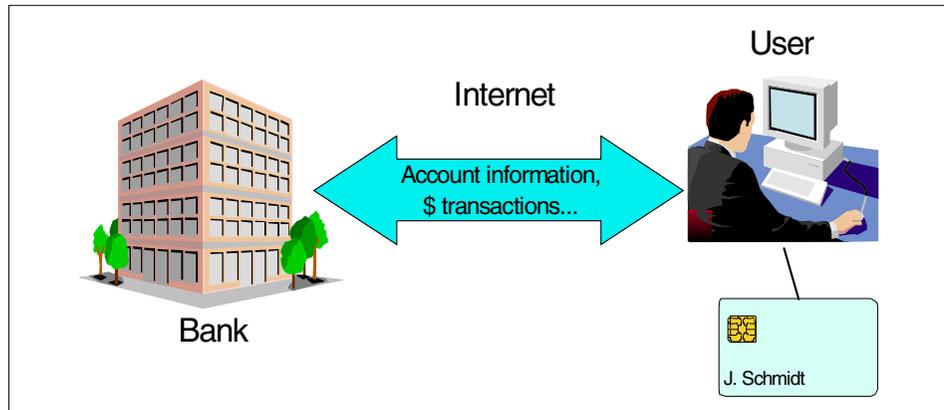


Figure 3-12 Banking example

Figure 3-12 illustrates the use of smart cards in a banking environment. The bank establishes a telebanking function over the Internet. The user can communicate with the bank using SSL with client authentication, using his or her keys.

3.3 Where to find more information

This chapter is a short introduction for PKI and related technologies. Obviously, it cannot cover all features and parts of PKI systems and processes; further information can be found in the following publications:

- ▶ Deploying Public Key Infrastructure, SG24-5512
- ▶ GPK Reference Manual (GemPlus 2001)
- ▶ Working with Business Process Objects for Tivoli PKI, SG24-6043
- ▶ Handbook of Applied Cryptography (A. Menzes, P van Oorschot, S. Vanstone - CRC Press,1996)

Some useful Web sites:

- ▶ Tivoli's Web site: <http://www.tivoli.com>
- ▶ Entrust's Web site: <http://www.entrust.com>
- ▶ Baltimore's Web site: <http://www.baltimore.com>
- ▶ RSA's Web site: <http://www.rsa.com>
- ▶ VeriSign's Web site: <http://www.verisign.com>

- ▶ Gemplus's Web site: <http://www.gemplus.com>
- ▶ Datakey's Web site: <http://www.datakey.com>



Part 2

WebSphere security



IBM WebSphere Application Server security

This chapter is an introduction to IBM WebSphere Application Server V4 Advanced Edition security.

It covers the security architecture and the basic security settings for the server, and serves as a quick overview of all the security features and services within WebSphere. Most of the features and options will be discussed in detail in the following chapters.

4.1 WebSphere security model

The IBM WebSphere Application Server V4 is a J2EE 1.2 compliant Java application server; it implements the required security services as they are specified.

The security components are essential parts of the application server architecture. The following description will give a high-level overview of these components.

4.1.1 Security architecture

The application server includes these components related to application security:

- ▶ Security server
- ▶ Security Collaborator
 - Web Collaborator
 - EJB Collaborator
- ▶ Security Policies
- ▶ Security Information

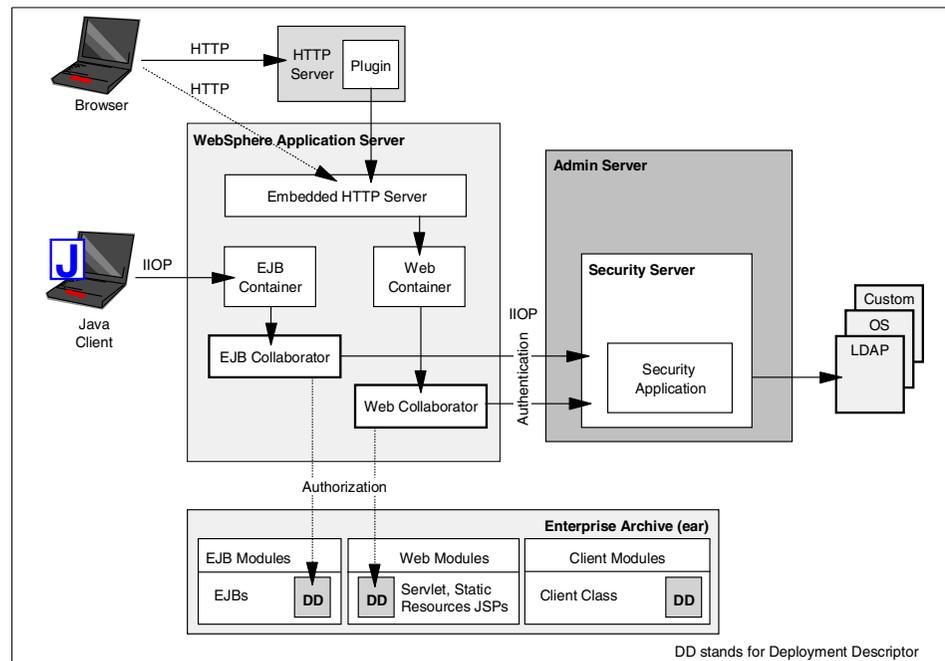


Figure 4-1 Overview of the security architecture

Note that in Figure 4-1, DD stands for Deployment Descriptor.

Security server

The Security server provides authentication services. It will, for example, use the included LTPA server to authenticate a user and to obtain a credential that can be used in the security context to represent a user identity. In this case, the Security server is a trusted third party for security policy and control of the applications.

The Security server provides the following services for Web containers and EJB containers:

- ▶ Authentication
- ▶ Authorization
- ▶ Delegation policies

Security collaborators

The Security collaborators reside in the application server process and are the major component for enforcing security constraints and attributes specified in the deployment descriptors. The Security collaborators also delegate the task of authentication to the Security server.

The following collaborators are identified:

- ▶ Collaborator in the Web container
- ▶ Collaborator in the EJB container

Web collaborator

The Web collaborator provides the following services:

- ▶ Checks the authentication if not provided
- ▶ Performs the authorization check
- ▶ Logs security tracing information

EJB collaborator

The EJB collaborator relies on the Secure Association Service (SAS) to authenticate Java client requests to enterprise beans. SAS provides message protection or encryption between clients and application server through IIOPS (RMI/IIOP over SSL). The Security collaborator works with the Security server to perform the following services for every remote method invocation of an enterprise bean:

- ▶ Check authorization.
- ▶ Support user registries.
- ▶ Log security tracing information.

- ▶ Set the *run-as* identity based on the delegation policy. The delegation policy determines the identity used when an enterprise bean invoke methods on another enterprise bean. The information about the delegation policy is stored in the Deployment Descriptor (ejb-jar.xml).

Security policies

Security attributes for enterprise beans and Web applications are specified in their deployment descriptors, which are XML files. The XML files contain much more than security information, but only security-related elements will be discussed in this book.

The security attributes are:

- ▶ Role and method permission
- ▶ Run-as mode or delegation policy
- ▶ Login configuration or challenge type
- ▶ Data protection (confidentiality and integrity) settings

The following diagram shows the object mappings for security and the relations between the objects. The * (asterisk) on one end of the relation means that many objects can be connected to that end (as it is used in the context of relational databases). For example, several users can be mapped to a group, and the same user can be in several groups; consequently, the relation between users and groups is many-to-many.

Users are assigned to roles directly or through groups. Methods are also assigned to roles. The role itself defines a set of permissions to access particular resources.

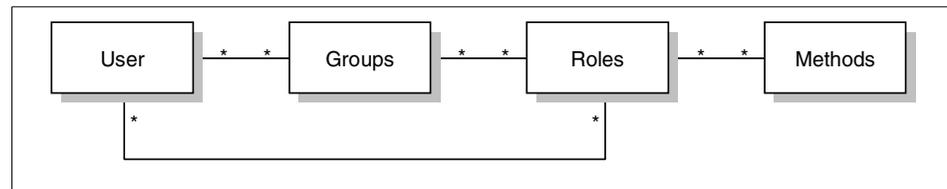


Figure 4-2 Role and method permission

Security information

Security information about WebSphere can be divided into two parts:

- ▶ Global security
- ▶ Application security

Global security is set for all applications running on the application server. It defines the type of registry and other security-related options. Global security information is coupled with the system management facility and, therefore, all the security information resides in XML configuration files (IBM WebSphere Application Server V4.0 Single Server) and in the database.

Application security can be defined and specified for each application; this means that the global settings can be overridden by application-specific settings. Application security is customized using the Application Assembly Tool, the Administrator's Console and the WebSphere Control Program (WSCP) tool. The Security server supplements the security runtime components (Security collaborator). The security runtime components acquire the system security configuration from the Security server; this includes such elements as the authentication mechanism, the user registry, etc.

A sample application security configuration can be found in Example 4-1.

Example 4-1 web.xml - XML configuration file

```
...
<web-app id="WebApp">
  <display-name>webbankWeb</display-name>
  ...
  <security-constraint id="SecurityConstraint_1">                                (1)
    <web-resource-collection id="WebResourceCollection_1">
      <web-resource-name>WebBank Transfer</web-resource-name>
      <url-pattern>/</url-pattern>
      <url-pattern>/webbank.html</url-pattern>
      <url-pattern>/TransferServlet</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint id="AuthConstraint_1">
      <description>All role for WebBank Transfer:+:</description>
      <role-name>AllAuthenticated</role-name>
    </auth-constraint>
    <user-data-constraint id="UserDataConstraint_1">
      <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
  </security-constraint>
  <login-config id="LoginConfig_1">                                          (2)
    <auth-method>FORM</auth-method>
    <realm-name>WebBank</realm-name>
    <form-login-config id="FormLoginConfig_1">
      <form-login-page>login.html</form-login-page>
      <form-error-page>login.html</form-error-page>
    </form-login-config>
  </login-config>
  <security-role id="SecurityRole_1">                                        (3)
```

```

        <description>A manager in the enterprise.</description>
        <role-name>Manager</role-name>
    </security-role>
    <security-role id="SecurityRole_2">
        <description>An employee in the enterprise.</description>
        <role-name>Employee</role-name>
    </security-role>
    <security-role id="SecurityRole_3">
        <description>All Authenticated in the enterprise.</description>
        <role-name>AllAuthenticated</role-name>
    </security-role>
    <security-role id="SecurityRole_4">
        <description>Everyone in the enterprise.</description>
        <role-name>Everyone</role-name>
    </security-role>
    <security-role id="SecurityRole_5">
        <description>Deny all access role</description>
        <role-name>DenyAllRole</role-name>
    </security-role>
    <env-entry id="EnvEntry_1"> (4)
        <description>the maximum limit the customer accounts
overdrafts</description>
        <env-entry-name>webbank/OverdraftValue</env-entry-name>
        <env-entry-value>5000</env-entry-value>
        <env-entry-type>java.lang.Integer</env-entry-type>
    </env-entry>
    ...
</web-app>

```

The following important tags are identified in the configuration file:

1. <security-constraint> - This defines a security constraint, meaning some URL patterns (for example: / or /TransferServlet) which have a specific method permission (for example: GET, POST) and can only be executed by a specific role (for example: AllAuthenticated).
2. <login-config> - This defines the type of authentication (basic, certificate, form based, digest).
3. <security-role> - This specifies the different security roles.
4. <env-entry> - A parameter is set which is not directly related to security but can be used for programmatic security.

More details can be found in Chapter 6, “Securing Web components” on page 105 and Chapter 7, “Securing EJBs” on page 135.

Important: Editing the EAR file directly can have some serious consequences, especially with respect to the security policy. In a server group configuration, the security information in the deployment descriptors within the EAR file may exist on more than one machine. You need to be very careful to keep all of the EAR files in sync.

4.2 IBM WebSphere Application Server security features

This section will discuss the IBM WebSphere Application Server security features and the provided security services.

The following topics will be discovered in detail:

- ▶ How to secure an application
- ▶ The WebSphere authentication model
- ▶ User registry
- ▶ Security Center
- ▶ Web Trust Association
- ▶ Securing only the Administrative server

4.2.1 How to secure an application

Applications are the highest set of components running under the application server. The process of securing an application with IBM WebSphere Application Server is described below.

Important: The whole application is as secure as its weakest point; this means that forgetting one step can have tremendous effects on security.

Table 4-1 helps to identify the storage locations (descriptors and configuration files) for the different application elements; it also provides information about the configuration tool.

Table 4-1 Overview of security information, storage location and tools

Security information	Storage location	Tool
Define business roles (whole application)	application.xml	AAT
Web resources	WAR DD web.xml	AAT
EJB resources	EJB DD ejb-jar.xml	AAT
Associate Principals	EAR DD ibm-application-bnd. xmi; Repository DB	AAT, AC
Configure global security authentication	Repository DB	AC

Table 4-2 Creating the security constraints for Web resources

Security information	Storage location	Tool
Define Web component authentication	WAR DD web.xml	AAT
Define security constraints and assign it to roles	WAR DD web.xml	AAT
Optionally, define Role Reference	WAR DD web.xml	AAT,AC
Secure static resources delivered by the Web server		

Table 4-3 Creating the method permissions for EJBs

Security information	Storage location	Tool
For each EJB, assign each of its methods to one or more roles	EJB DD ejb-jar.xml	AAT
Optionally, set up Security Role References	EJB DD ejb-jar.xml	AAT, AC
Configure the delegation policy - Run-As	EJB DD ibm-ejb-jar-ext.xmi	AAT
Run-As Mapping	Repository DB	AC

In the above tables, AAT refers to *Application Assembly Tool*, AC refers to *Administrator's Console*, DD refers to *Deployment Descriptor*.

WebSphere V4 Advanced Edition supports the multi-server environment, which requires centralized management. Since each container on each server has its own deployment descriptor (DD) in a file system, there is a need for a common repository, which is a database in this case. The database and the deployment descriptors are synchronized automatically by the system, but there are certain situations where the modification in the deployment descriptor is not picked up by the repository during runtime.

For example, the role mapping stored in the repository is based on the role information in the deployment descriptors at the point of deployment. Adding new roles after deployment can only be done by redeploying the application.

The following steps summarize how to assign security to an application using the Application Assembly Tool (AAT).

1. Define business roles based on the rights of the people.

In Table 4-4, some roles other than the roles from the Webbank example are defined, to make the *role* concept clear.

Table 4-4 Role definition

Role	Description
Manager	A manager can create an account and cancel a business contract.
Employee	An employee has access to the company resources, but cannot make transfers above a certain amount.
Consultant	A consultant has access to the application, can access information, but cannot perform transfers.

Roles created within the EJB module are also available in the Web module and vice-versa.

Figure 4-3 shows the AAT with the defined roles.

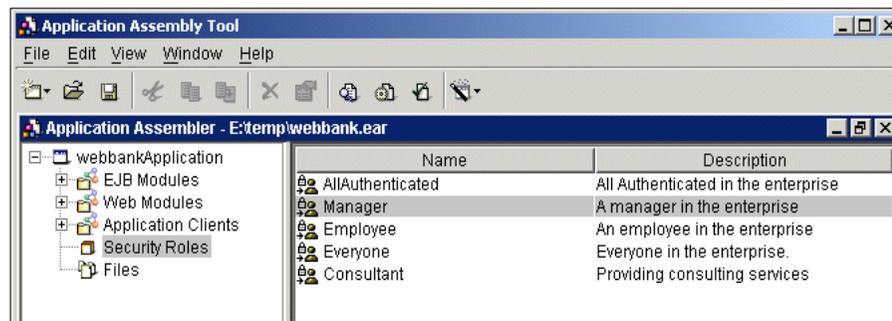


Figure 4-3 AAT with defined roles

2. Create the security constraints for Web resources.
3. Define the Web component authentication for the Web module (*.war). The login authentication challenge type and constraints on transports are specified at the Web application level. For more information about Web component security, refer to Chapter 6, “Securing Web components” on page 105.
4. Define security constraints and assign them to roles. Security constraints are restrictions for resources of a specific URI pattern.
 - Specify the URL patterns which belong to a Web resource collection (Servlet, JSP, HTML...).
 - Assign an HTTP method permissions (GET, POST...) to the Web resources.
 - Assign at least one role to the Web resources collection.

In Figure 4-4, we show you the structure of a Web module from a security standpoint, using the Webbank sample application.

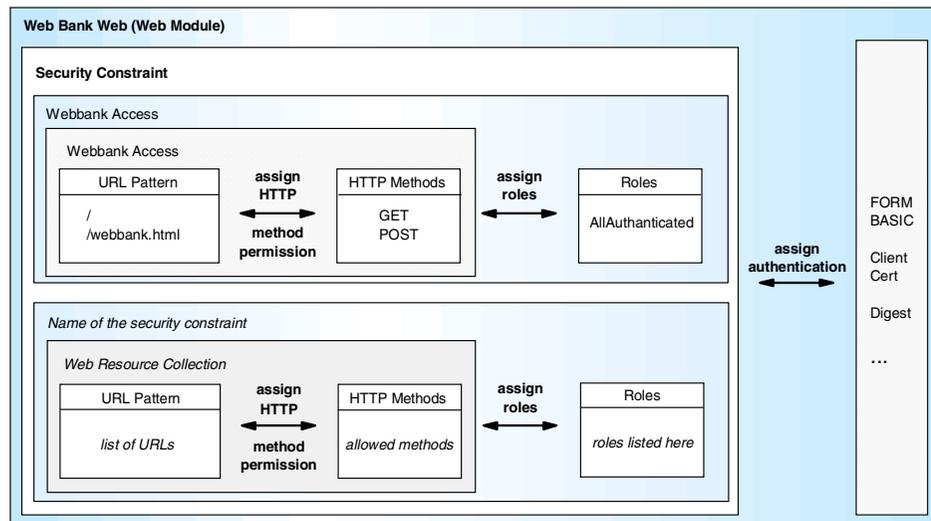


Figure 4-4 Security constraints in a Web module

Figure 4-5 shows the AAT with the defined security constraints.

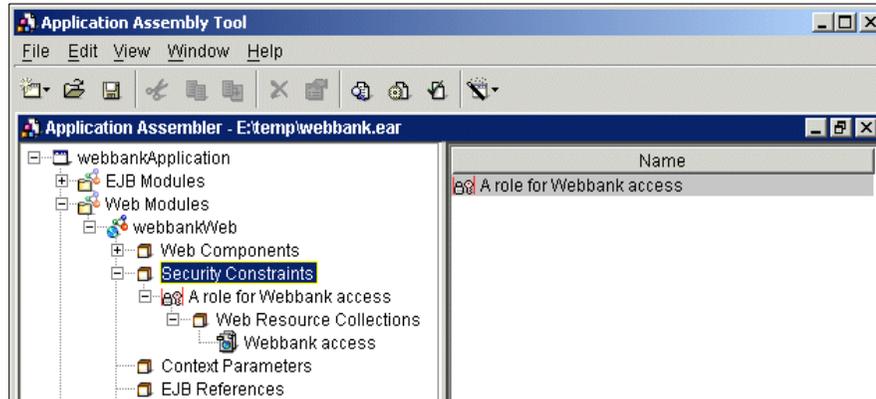


Figure 4-5 Security constraints (AAT)

5. Security Role References can be set up as an option. These elements allow the Web resources to perform programmatic security checking, if this is desired. These references are associated with local or internal role names that developers might want to use in their code (for example: programmatic security). An example allowing the CEO of a company special rights, which are defined in the implementation.
6. You may also want to secure static resources (HTML, image, audio, video files), which are delivered by the Web server. For more details, see Chapter 6, “Securing Web components” on page 105.

Note: Most parameters in the *web.xml*, apart from `<security-constraints>`, `<security-role>`, and `<login-config>` can be changed without restarting the server.

Changing the listed parameters requires you to restart the server.

7. Create the method permission for the EJBs. For more information about EJB security, refer to Chapter 7, “Securing EJBs” on page 135.

Figure 4-6 depicts the structure of the EJB module from a security standpoint, using the Webbank example.

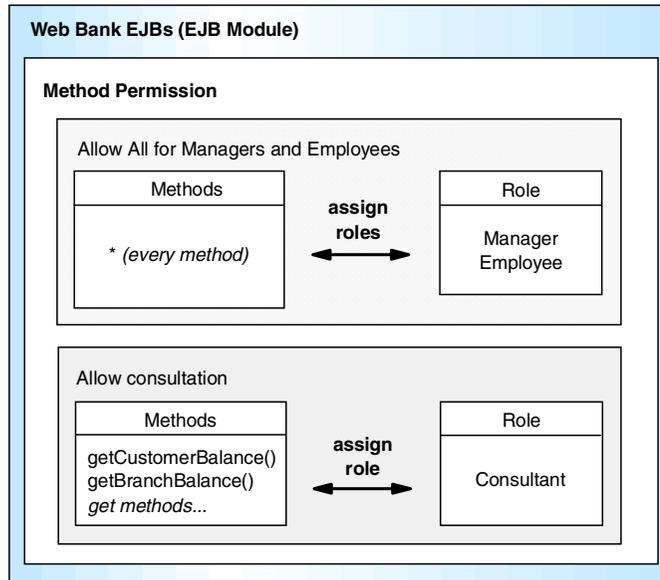


Figure 4-6 Method permission

8. Set up Security Role References as an option. These elements enable the EJB to perform programmatic security checking, if this is desired. These references are associated with local or internal role names that developers might want to use in their code.
9. Configure the delegation policy (Run-As Mode) which determines the identity to use if the Enterprise bean invokes methods on any other Enterprise bean. The delegation policy is specified in the *ejb-jar.xml* deployment descriptor. The following three options are available:
 - *Client identity*, which is the identity of the caller.
 - *System identity*, which is the identity of the intermediary.
 - *Specified identity*, which is based on a particular role, named in the delegation policy.

Figure 4-7 shows the AAT with the window where the application assembler can define the Run-As Mode for the method.

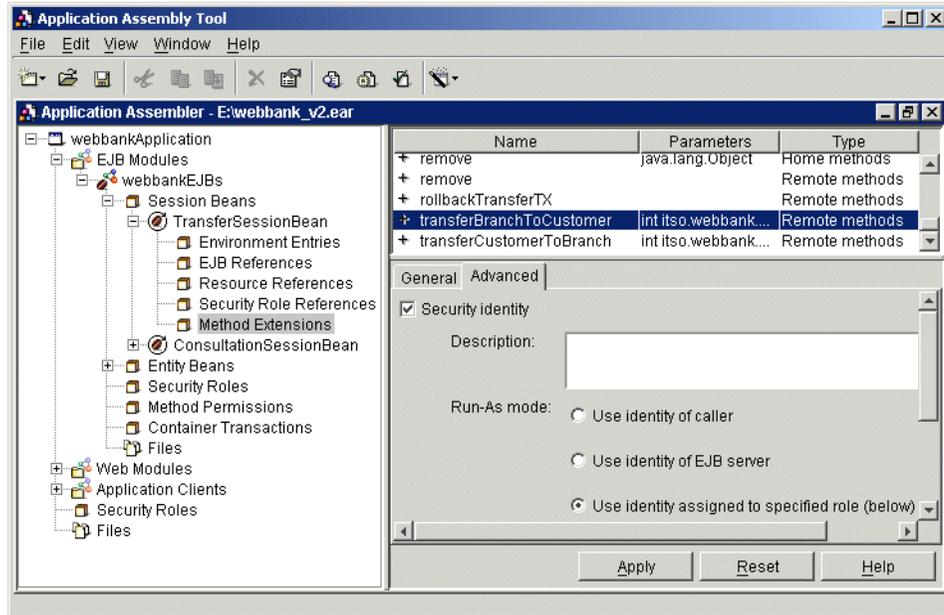


Figure 4-7 EJB Run-As Mode definition

10. Select the *Run-As* role from a specific Enterprise application and assign it to a role from the user registry. This can be done using the Administrator's Console (**Security Center -> Run As Role Mapping**).

For more information, refer to Chapter 7, "Securing EJBs" on page 135.

11. Associate Principals, namely users and groups with roles. There are three special roles besides the users and groups, called special subjects: *AllAuthenticated*, *Everyone* and *DenyAllRole*.

12. Start the Administrator's Console and open the Security Center.

13. Configure global security authentication. There are three different user registries for authentication:

- LDAP
- OS
- Custom User Registry

If you have a certificate-based authentication (LDAP, Custom User Registry), you may also want to provide your customer with a certificate; find more details in "CustomRegistry SPI" on page 206.

14. Configure the SSL settings using the Administrator's Console (AC). This means using SSL:
- HTTPS is used between the Web server plug-in and the application server
 - There is a secure ORB between the client ORB and the server ORB.
 - LDAPS provides secure communication between the Administrative Server and the LDAP registry used for authentication. This feature is available only in IBM WebSphere Application Server Advanced Edition.

4.2.2 WebSphere authentication model

This section discusses the IBM WebSphere Application Server's authentication model. Usually, you apply security to a given user which has a user ID and belongs to a group.

There are four possible authentication mechanisms defined in the servlet specification. For details, see *Java Servlet Specification V2.2* at <http://java.sun.com/products/servlet/2.2>.

- ▶ HTTP Basic authentication
- ▶ HTTP Digest authentication (not supported in this release of IBM WebSphere Application Server)
- ▶ HTTPS Client Certificate authentication
- ▶ Form-Based authentication

All authentication mechanisms except Client Certificate authentication have the problem of target server authentication. The Form-Based and Basic authentication have, in addition, the problem of the password not being encrypted. Additional mechanisms are necessary to avoid authentication and confidentiality problems; secure transport (SSL) can be used, but network-level security such as IPSEC protocol or VPN can also work.

Basic authentication

In this authentication, the Web browser presents a dialog window requiring the user to enter a user ID and password when attempting to access a protected Web resource. After the user provides the ID and password, the security service validates them against the user registry. Basic authentication is not a secure protocol for authentication. The password is only encoded in simple base64 and therefore not secure. The target server is not authenticated either, which also poses a security risk. To avoid authentication and confidentiality problems, secure transport (SSL) can be used in combination with network-level security such as IPSEC protocol or VPN.

Example 4-2 web.xml - Basic authentication

```
<login-config id="LoginConfig_1">
  <auth-method>BASIC</auth-method>
  <realm-name>WebBank</realm-name>
</login-config>
```

Digest authentication

In Digest authentication, a user ID and password are transmitted as in Basic authentication, but the password is encrypted. This mechanism is more secure than the Basic authentication with base64 encoding. Currently, this authentication is not widely used and the servlet containers are not required to support it. Because of that, this authentication mechanism is not supported by WebSphere. The developers can program additional security with JCE, but they must support it themselves.

Example 4-3 web.xml - Digest authentication

```
<login-config id="LoginConfig_1">
  <auth-method>DIGEST</auth-method>
  <realm-name>WebBank</realm-name>
</login-config>
```

Important: The digest authentication method is not supported in this release of IBM WebSphere Application Server.

Client authentication

This authentication mechanism requires the client to possess a Public Key certificate. The identity in the digital certificate is mapped to an entry in either the LDAP registry or a custom user registry. Certificates are transmitted from browser to Web server over HTTPS.

Example 4-4 web.xml - Client authentication

```
<login-config id="LoginConfig_1">
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>WebBank</realm-name>
</login-config>
```

Form-based authentication

This authentication mechanism permits a site-specific login through an HTML page or a JSP form. The password is not encrypted and the target server is not authenticated, which is also a security risk. To avoid authentication and confidentiality problems, secure transport (SSL) should be used.

Example 4-5 web.xml - Form-based authentication

```
<login-config id="LoginConfig_1">
  <auth-method>FORM</auth-method>
  <realm-name>WebBank</realm-name>
  <form-login-config id="FormLoginConfig_1">
    <form-login-page>login.html</form-login-page>
    <form-error-page>error.html</form-error-page>
  </form-login-config>
</login-config>
```

As in the defined servlet specification, the login form must contain fields for entering a user ID and password. For more information about how to use form-based login, refer to Chapter 6, “Securing Web components” on page 105.

The deployment should also contain the entry for the login form and the error page which can be specified in Application Assembly Tool (AAT).

When a user accesses a protected Web resource, the container checks the user authentication. If the user is not authenticated, the following steps are taken:

1. The login form, which is associated with the security constraint, is sent to the client. The URL path which has triggered the authentication is stored by the WebSphere servlet container.
2. The client sends the filled-out form back to the server and the container attempts to authenticate the user by the defined registry.
 - a. If the authentication fails, then the error page provided earlier is sent and the status code of the response is 401.
 - b. If the authentication succeeds, then the user is authorized and the requested protected resource is transmitted to the client.

Form-based authentication also has the problems of plain text transmission and target server authentication. Again, secure transport should be used.

Because the container supports Single Sign-On, the user does not need to reauthenticate to other protected resources. For more information, refer to Chapter 14, “Single Sign-On” on page 393.

Configuration using AAT

To define the login authentication challenge type, the data in web.xml is configured using the Application Assembly Tool (AAT).

The application (application.xml) consists of Web modules (web.xml) and EJB modules (ejb-jar.xml); the configuration can be found in the corresponding deployment descriptor files.

The authentication data of Web modules can be read at the Web module within AAT. The authentication data of the Web modules can be configured using a pull-down menu in the GUI, as shown in Figure 4-8.

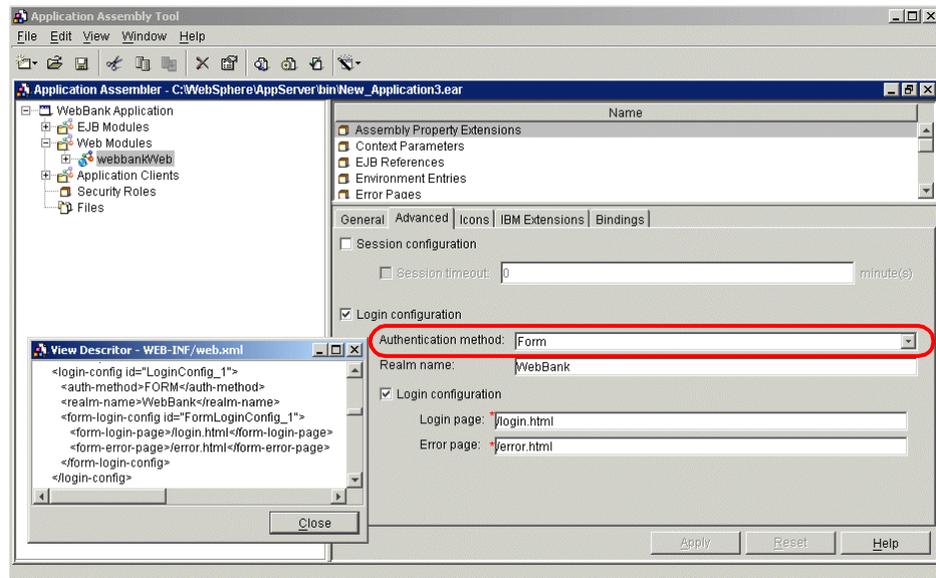


Figure 4-8 AAT authentication data

Within the *Authentication method* listbox, the following choices are available:

- ▶ All methods
- ▶ Basic
- ▶ Digest
- ▶ Form
- ▶ Client cert

The child window shown in Figure 4-8 shows the deployment descriptor. The login authentication challenge type and constraints on transports are specified at the Web application level (WAR's Deployment Descriptor) inside the `<login-config>` element.

The configuration using WebSphere Studio Application Developer (WSAD) is documented in “Configuring Web module security using WebSphere Studio Application Developer” on page 127.

The settings defined on the Web module are stored in the `web.xml` deployment descriptor. Using the WebSphere Studio Application Developer for these settings brings up the same result as using the Application Assembly Tool.

4.2.3 User registry

The user registry is a repository that contains users and groups. This applies to both human users of the system and applications validated through the user registry.

The administrator can have users or groups authenticated against the local operating system user registry (such as Windows NT User Manager program) or an LTPA (LDAP or Custom User Registry).

The following section shows the possible user registries for authentication available in WebSphere. The local operating system and LDAP are fully supported by WebSphere; the custom user registry is the developers' responsibility. To provide an overlapping authentication mechanism (OS and LDAP), the custom registry should be used.

The following list shows the available authentication mechanisms:

- ▶ Local operating system:
 - Windows NT(Domain, WorkGroup)
 - Windows 2000
 - AIX
 - Solaris
 - HP-UX
 - Linux
- ▶ Lightweight Third Party Authentication (LTPA):
 - LDAP (only for Advanced Edition)
 - Netscape Directory Server
 - Domino 4.6, 5.0
 - IBM Secure Way Directory
 - Windows 2000 Active Directory
 - Custom User Registry (only for Advanced Edition)

Operating systems support Basic and Form-based authentication, whereas LDAP and Custom user registries support both password- (basic, form) and certificate-based authentication. LTPA is not supported by OS registries.

Table 4-5 on page 51 summarizes the supported authentication possibilities for the different user registries.

Table 4-5 Authentication mechanisms in WebSphere

	OS (Unix)	OS (NT)	LDAP	Custom
Basic	Authentication using system calls	Authentication using security Access Manager through system calls	An LDAP search is performed.	A password check is performed against the custom registry.
Form-Based	Authentication using system calls	Authentication using security Access Manager through system calls	An LDAP search is performed.	A password check is performed against the custom registry.
Certificate	N/A	N/A	The certificate content is a credential mapped to an LDAP entry (based on trust of the Web server).	The certificate content is a credential mapped to a custom entry (based on trust of the Web server).
Digest	N/A	N/A	N/A	N/A

Table 4-6 shows the authentication mechanism emphasizing the type of the client.

Table 4-6 Authentication for Web and Java clients

Challenge type	Authentication mechanism	User registry	Client
None	None	None	Web / Java
Basic	LTPA	LDAP	Web / Java
		Custom	Web / Java
	OS	OS	Web / Java
Certificate	LTPA	LDAP	Web
		Custom	Web

Challenge type	Authentication mechanism	User registry	Client
Form Login	LTPA	LDAP	Web
		Custom	Web
	OS	OS	Web

The Certificate-based challenge only has LTPA. At LTPA, you have the choice between LDAP and Custom User Registry. If Basic authentication is used, the LTPA and OS can be used from a Web client and a Java client, whereas a certificate can only be used by a Web client.

4.2.4 Security Center

This section provides information about the Security Center, which is part of the Administrator's Console (AC).

General tab

Switch to the General tab to enable or disable security. To enable security, the **Enable Security** checkbox must be selected, otherwise, other security settings specified will be disregarded.

Note: The global security settings (enabled/not enabled), authentication mechanism, and user registry settings are stored in the Admin Repository.

This page also contains an option for setting a security cache time-out. The security system caches authentication lookup information it receives from the user registry or directory service. This field specifies how long (in seconds) the authentication information will be cached for performance reasons.

Figure 4-9 shows the General tab in the Security Center, where you can specify the general security settings.

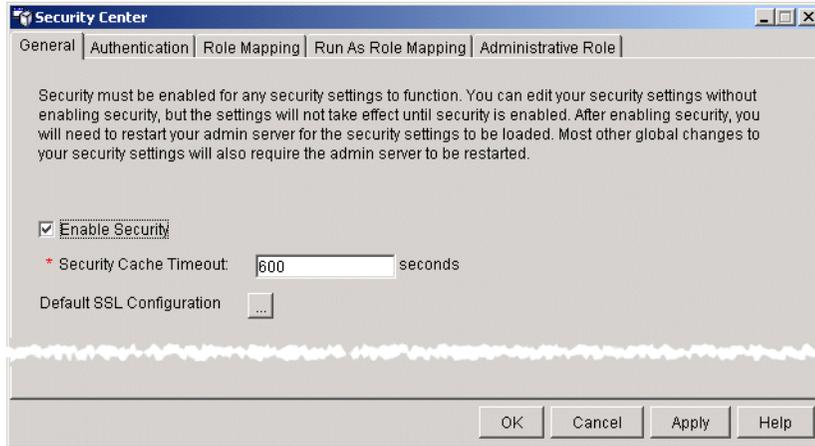


Figure 4-9 Security Center - General tab

The General tab provides an option: *Default SSL Configuration*, where the SSL settings can be configured; for more information, refer to “Configuring WebSphere to use your own keyring” on page 229.

Authentication tab

Switch to the Authentication tab in order to specify how to authenticate the information presented by users trying to access an application or resources. The information depends on the type of the registry.

The user ID under which the server runs, for security purposes, should point to a valid user in the OS/LDAP/Custom user registry. The password must correspond to the Security server ID. Additional information must be provided depending on the user registry (see Figure 4-10 on page 54).

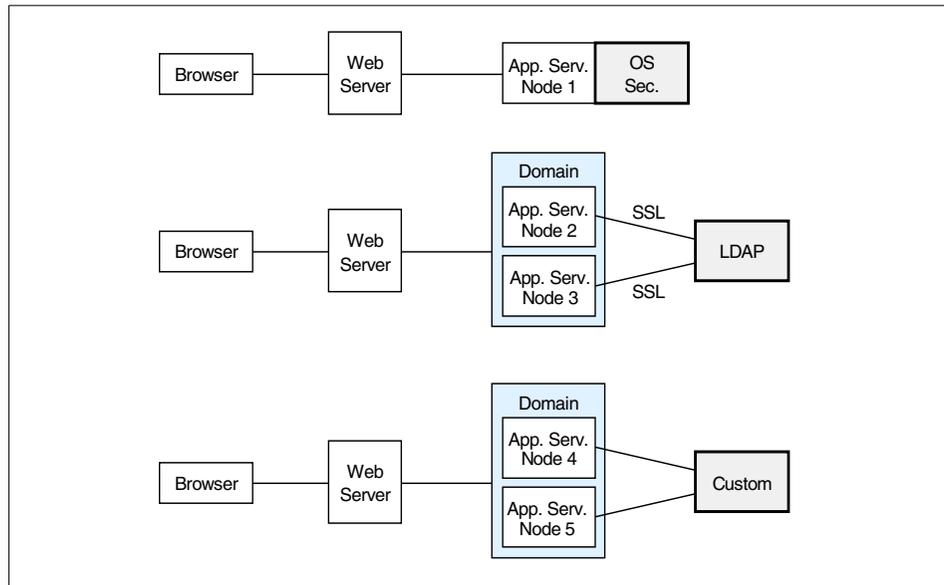


Figure 4-10 Different types of user registries

Note that the local operating system user registry is intended for single machine and single application server environments.

As seen before, authentication can be customized using the Administrator's Console. In the following sections, each user registry is explained in detail.

Local operating system

The local operating system can be used to authenticate the application user. The following important facts should be noted:

- ▶ Local registries are limited to single-machine or Windows NT domain-controller environments and a single application server. IBM WebSphere Application Server does not support multiple node, multiple application servers or secure delegation when the Local registry is used as the user registry.
- ▶ NT Domain and Workgroups are supported, not Trusted Domains.
- ▶ If a machine is a member of a Windows domain, both the domain user registry and the machine's local user registry are used for authentication and security role mapping.
- ▶ The domain user registry takes precedence over the local user registry.
- ▶ Security roles should always be mapped to domain users and not to local users.

- ▶ Do not use an account the name of which matches the name of your machine or Windows domain.
- ▶ It is recommend that you use the user registry where WebSphere is local to your node.

If a user is registered in the domain or locally with the same password, you may encounter problems during the authentication.

How to set local operating system authentication mechanism:

Use the following steps to select the local operating system as the WebSphere security authentication mechanism:

1. Create the local operating system user which should have access to the administrative server within Windows 2000.
2. Open Computer Management in Windows.
3. Select the **Users** folder (**System Tools -> Local and Users and Groups -> Users** folder).
4. Create a new user by selecting **Action -> New User** from the main menu. Create the WebSphere application users and an administrator user for the administrative server.
5. You might want to use groups instead of individual users for role mapping. Assigning groups to roles is more flexible than assigning individual users. To create a new user, select the **Users** folder (**System Tools -> Local and Users and Groups -> Users** folder) then **Action -> New User** from the main menu.
6. Figure 4-11 on page 56 shows Windows's user management interface.

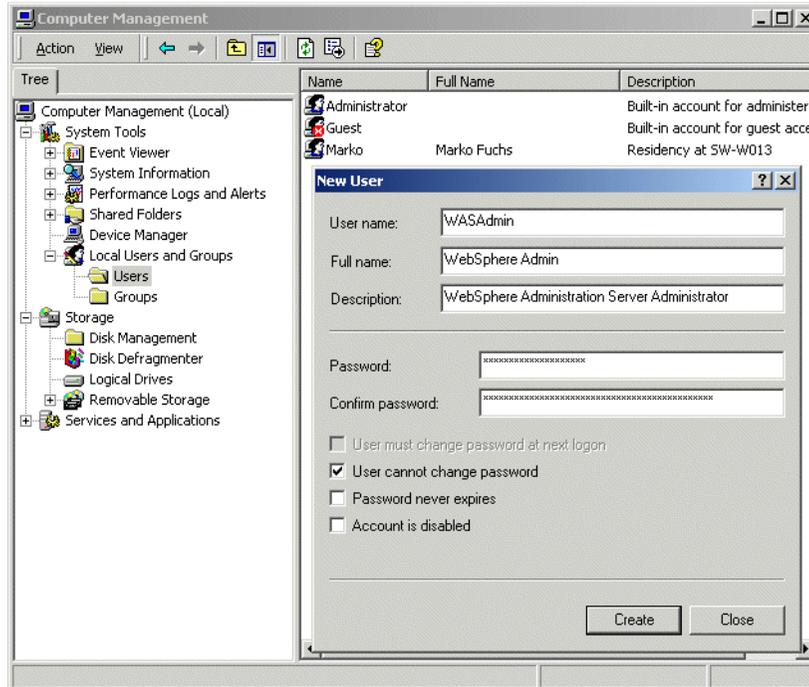


Figure 4-11 Creating a user in Windows 2000

7. Configure the Admin Server. Open the Administrator's Console which is, by default, found by selecting **Start -> Programs -> IBM WebSphere -> Application Server V4.0 AE -> Administrator's Console**.
8. Select **Console -> Security Center**. Switch to the **Authentication** tab and choose **Local Operating System** as the Authentication Mechanism.
9. Set the Security server ID and password to the local operating system user ID and password you want to use to access the administrative server. Verify the user ID and password against the local operating system by clicking **OK**.

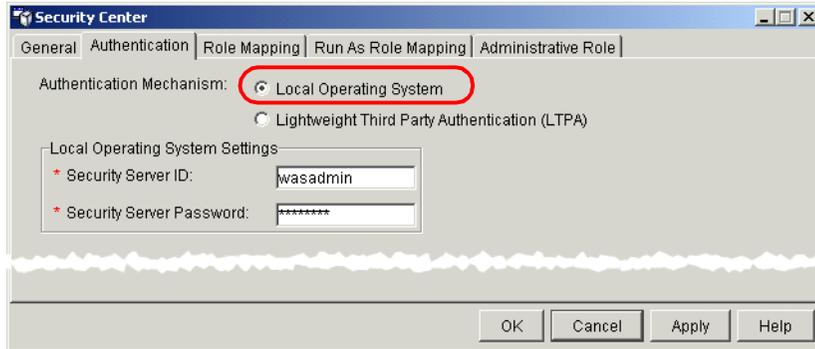


Figure 4-12 User Registry Operating System

10. Make sure that security is enabled at the general tab.
11. Restart the administrative server by right-clicking the node in the Administrator's Console, then select **Restart**.

LDAP

WebSphere supports two LTPA authentication mechanisms: LDAP and Custom User Registry. This section discusses the LDAP option. The following important facts should be noted:

- ▶ This is not available for WebSphere Single Server Edition.
- ▶ The user should *not* be a root DN or administrator DN because it is unnecessary to expose the root password.
- ▶ You may want to secure the connection between the application server and LDAP.

LDAP authentication can be set to use one of the following authentication mechanisms:

- ▶ Password based Authentication
- ▶ Client Certificate Authentication

Password-based

Refer to Section 16.1.3, “Configuring WebSphere to use the SecureWay Directory Server” on page 478.

Certificate-based

This section discusses the authentication mechanism using LDAP with client side certificates.

Figure 4-13 depicts the security flow using client side certificates with IBM WebSphere Application Server.

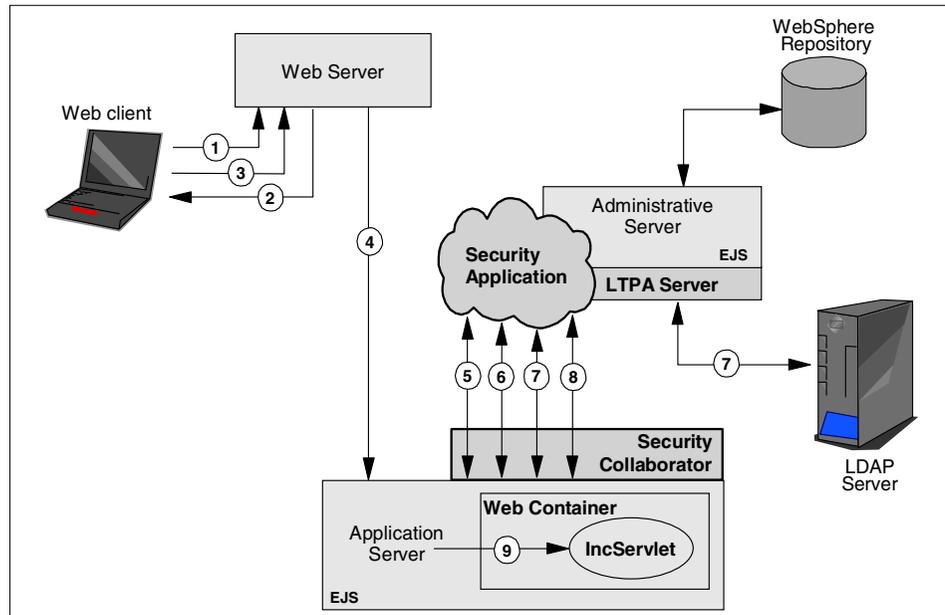


Figure 4-13 Security flow using certificates

The security flow of client authentication using certificates is as follows:

1. The request for the IncServlet servlet comes from the browser user to the Web server.
2. The Web server determines that the request is for a resource that requires client certificate authentication, so it challenges the browser to return a certificate.
3. The browser recognizes the challenge and returns the certificate.
4. The Web server authenticates the client certificate. It then determines that it does not control the servlet request, so it passes the client certificate along with the request to the application server.
5. The application server determines that it controls the servlet and, through the security collaborator, determines that the servlet is secure.
6. The application server, through the security collaborator, determines that the required authentication method is a client certificate.
7. Using the security collaborator, which in turn works with the security application, the application server determines the credentials of the certificate. The LTPA server component of the security application works with

the LDAP server to perform a credential mapping of the certificate to the contents of the LDAP directory.

8. Using the security collaborator, the application server determines that the user is authorized to access the servlet being requested.
9. The application server invokes the servlet for the user.

For more information about certificates, refer to Chapter 3, “Security certificates” on page 13; for information on how to set up security using certificates, refer to Chapter 11, “Administering WebSphere Security” on page 215.

Custom User Registry

This section describes the case when LTPA is set and a Custom User Registry is used for client authentication.

The following important facts should be noted:

- ▶ This is not available for Advanced Single Server Edition.
- ▶ You may want to secure the connection between the application server and the Custom User Registry.

Using a custom registry is not natively supported by IBM WebSphere Application Server. The application server needs a class that implements the custom registry interface by providing code for each required method. User and group information can be stored in databases, files, directory servers, etc.

Note: The following sample application should only show the features of the custom registry. The provided sample application should never be used in production, for performance and scalability reasons.

Configuration using the Administrator's Console

The following steps will guide you through the process of configuring your Custom User Registry for WebSphere:

1. Open the Administrator's Console and select **Console -> Security Center**.
2. Switch to the **Authentication** tab and choose **Lightweight Third Part Authentication (LTPA)** as the authentication mechanism.
3. The authentication panel will change. Choose **Custom User Registry**; the window will show the settings for this type of registration.
4. Set the security server id and password to the user ID and password you want to use to access the administrative server. The following settings have to be entered as well:
 - a. **Custom User Registry Classname:** the name of the class file which implements the custom registry code.

- b. You may also want to set Special Custom Settings, where you can specify settings for the Custom User Registry. In our example, the users.props and groups.props file had to be configured, as shown in Figure 4-14.
5. Click **Apply** in the Security Center.
6. Make sure that security is enabled under the General tab.
7. Restart the administrative server by right-clicking the node in the Administrator's Console, then select **Restart**.

Sample Custom Registry

This sample will use the File Registry sample provided in the WebSphere Advanced Edition V4 Infocenter at <http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html> under Section 5.2, "Introduction to custom registries." You can find the entire description of the sample in the mentioned section together with the source code. Follow the steps from the Infocenter to create the necessary files and compile the required code for the sample.

Figure 4-14 on page 61 depicts the Security Center with the settings for Custom User Registry and a child window, in which Special Custom Settings are specified.

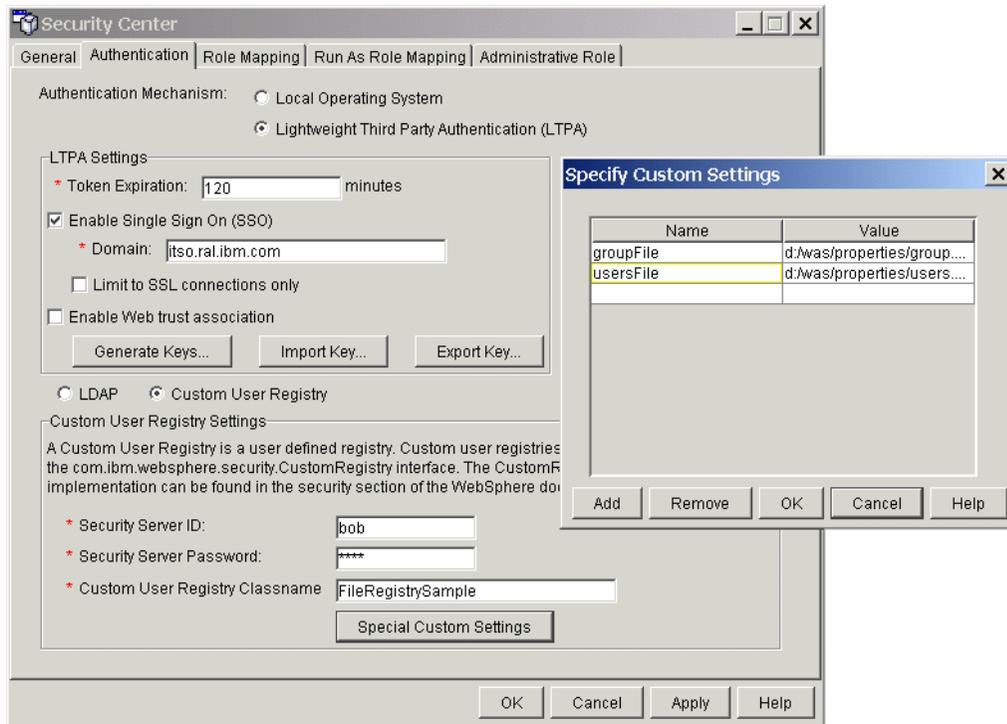


Figure 4-14 Custom Registry

The settings for the sample are:

1. Security server ID: bob
2. Security server Password: bob1
3. Custom User Registry Classname: FileRegistrySample
4. Click **Special Custom Settings**, then add the following two lines:

Table 4-7 Property files

Name	Value
groupFile	d:\was\properties\groups.props
usersFile	d:\was\properties\users.props

5. Click **OK** in the Specify Custom Settings child window.
6. Click **Apply** in the Security Center.
7. Make sure that security is enabled under the General tab.

8. Restart the administrative server by right-clicking the node in the Administrator's Console, then select **Restart**.

For more information about Custom User Registry SPI, refer to "CustomRegistry SPI" on page 206.

4.2.5 Web Trust Association

IBM WebSphere Application Server can authenticate incoming user requests, as we have seen before. In some scenarios, such as with Web-based applications, it is often desirable to delegate this work to another process. This process is typically a Reverse Proxy Security Server (RPSS).

In order to perform the delegation, a trusted relationship between the application server and the proxy must exist. This means that the proxy server authenticates the client and the application server accepts this because it trusts the proxy. IBM WebSphere Application Server applies its authorization policies to the requests.

Enabling Web Trust Association

The following steps will show you how to enable Web Trust Association in WebSphere.

1. Open the Administrator's Console and select **Console -> Security Center**.
2. Switch to the **Authentication** tab
3. Select the checkbox **Enable Web trust association** in the LTPA Settings pane (see Figure 4-15).
4. Click **Apply** in the Security Center.
5. Make sure that security is enabled under the General tab.
6. Restart the administrative server by right-clicking the node in the Administrator's Console, then select **Restart**.

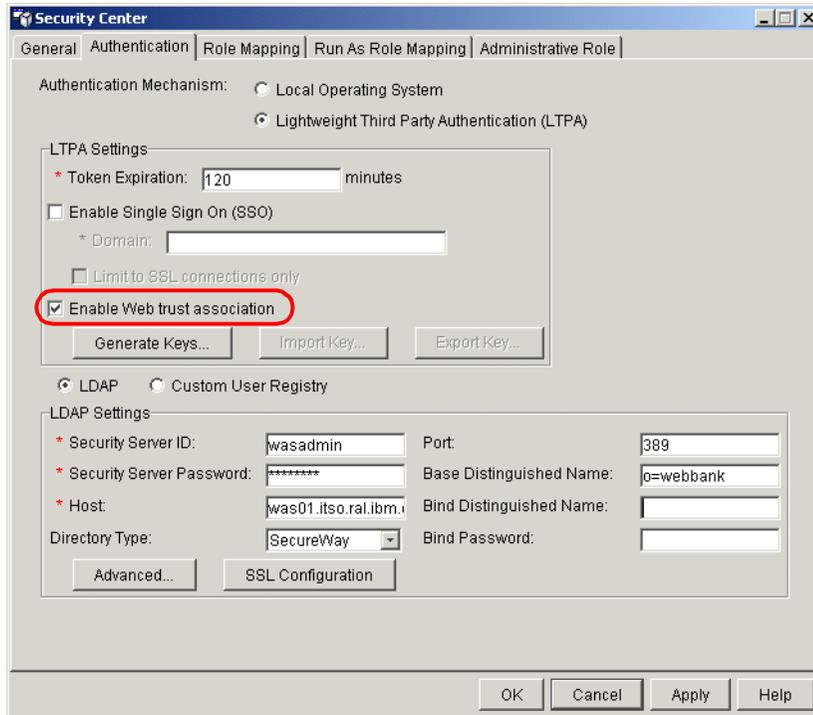


Figure 4-15 Enabling Web Trust Association

Usually, the user ID and password are sent in the HTTP header. By setting the flag, the user ID and password from *iv_user* and *iv_password* are used. If the flag is not set, then the authentication data is ignored, because the WebSEAL ID and password are used.

For more information about the Web Trust Association, refer to “Trust Association Interceptor SPI” on page 213; for information on Web Trust Association in relation with Tivoli Policy Director, refer to Chapter 13, “Policy Director” on page 353.

4.2.6 Securing only the Administrative Server

This section will provide information about how to secure the Administrative server within WebSphere without securing all the application servers.

1. Start the Security Center and enable security (General tab)
2. Restart the Administrative Server so that changes may take effect.
3. The next time the Administrator’s Console opens, the administrator will be prompted to log in, using the user ID and password.

When you enable WebSphere security, only the Administrative Server (see step 6 in “How to secure an application” on page 39) is protected. The application server(s) on the node is also protected if security constraints are defined. With WebSphere V4.0, you can turn off security for selected application servers on a node. This feature can be used to protect the Administrative Server by avoiding the overhead of secure encryption on application server communications. This is, of course, only suitable for application servers that do not require WebSphere security protection.

Disabling security in the Administrative Server

Security information (global security settings (enabled/not enabled), authentication mechanism, user registry settings) is stored in the Admin Repository. By resetting the information, the Administrative Server is made accessible again, which could be used for password recovery, for example. The following steps will help to disable security in the Administrative Server.

Note: You should always use the Administrator's Console Security Center to disable security.

1. Open the `sas.server.props` file under the `<WebSphere install path>\properties` directory in a text editor. According to the type of registry that has been chosen, the following settings are shown in the file:

In the case of LDAP registry:

Example 4-6 LDAP Registry in the sas.server.props file

```
com.ibm.CORBA.principalName=itsohost.ra1.ibm.com \:389/wasadmin
com.ibm.CORBA.securityEnabled=true
com.ibm.CORBA.loginUserId=wasadmin
com.ibm.CORBA.authenticationTarget=LTP
```

In the case of Custom User Registry:

Example 4-7 Custom User Registry in the sas.server.props file

```
com.ibm.CORBA.principalName=customRealm/wasadmin
com.ibm.CORBA.securityEnabled=true
com.ibm.CORBA.loginUserId=wasadmin
com.ibm.CORBA.authenticationTarget=LTPA
```

In the case of Local Operating System Security:

Example 4-8 Operating System Security in the sas.server.props.file

```
com.ibm.CORBA.loginPassword={xor}Pg\=\=  
com.ibm.CORBA.principalName=WTRNTDM/wasadmin  
com.ibm.CORBA.securityEnabled=true  
com.ibm.CORBA.loginUserid=wasadmin  
com.ibm.CORBA.authenticationTarget=LOCALOS
```

2. Stop the Administrative Server.
3. Delete the sas.server.props.future file in the <WAS_HOME>\properties\ directory.

Note: If this file is present when the server restarts, information in the sas.server.props.future file is copied into the sas.server.props file, overwriting your changes.

4. Open the sas.server.props in the <WAS_HOME>\properties\ directory and set the *com.ibm.CORBA.securityEnabled* entry to false; then set *com.ibm.CORBA.securityEnabled* to false also.
5. Update EJSADMIN.SECURITYCFG_TABLE in the administrative repository to set security to 0 (zero).

The following is a script showing how to set the value in the database to 0 (zero).

Example 4-9 Database script to disable security

```
db2cmd  
db2 connect to was401 user db2admin using db2admin  
db2 update ejsadmin.securitycfg_table set securityenabled=0
```

Important: Secure the application immediately after the recovery.

Note: To recover the password, the user directory (for example: LDAP) could also be used. If your user directory is not accessible, then this approach will not help. The *sas.server.props* file is not secured in the file system. You may want to secure the file system to avoid having someone set the file to false: *com.ibm.CORBA.securityEnabled=false*.

Additional administrators to access the Administrative Server

Only one user specified in the Administrator's Console has access to the Administrative Server, by default. Usually, there are more administrators who need access to the system.

1. Start the Administrator's Console and open the Security Center.
2. In the Security Center, select the **Administrative Role** tab.
3. Select **AdminRole**, click **Select**; a new window opens where you can select the checkbox **Select Users and Groups**.
4. Now you can supply a pattern and search for users and groups in your user registry (defined in Authentication); for example, * (asterisk) will show all the users and groups.
5. Choose, in the left pane, one of the users you want to add (for example, **wasadmin**) as an administrator. Click **Add** (you will see the added user in the right pane).
6. Close the Select Users/Groups child window by clicking **OK**.
7. Click **Apply** in the Security Center.
8. Restart the Administrative Server by right-clicking the node in the Administrator's Console, then select **Restart**.
9. Check the settings by closing the Administrator's Console then restarting it and logging in as a new administrator.

4.3 WebSphere security and the operating environment

In this section, we discuss how WebSphere security relates to the security provided by your operating system and by Java.

IBM WebSphere Application Server security sits on top of your operating system security and the security features provided by other components, including the Java language, are shown in Figure 4-16.

- ▶ **Operating system security** is used to secure sensitive files in the WebSphere product installation and to authenticate users using the operating system user registry.
- ▶ **Java language security** is provided through the Java Virtual Machine (JVM) used by WebSphere and the Java security classes.
- ▶ **CORBA security** is used for inter-application communication between secure ORBs invoked using the Secure Association Service (SAS).

- ▶ **J2EE security** uses the security collaborator to enforce J2EE-based security policies and support J2EE security APIs.
- ▶ **WebSphere security** relies on and enhances all of the above. It enforces security policies and services in a unified manner for Web and EJB resources.

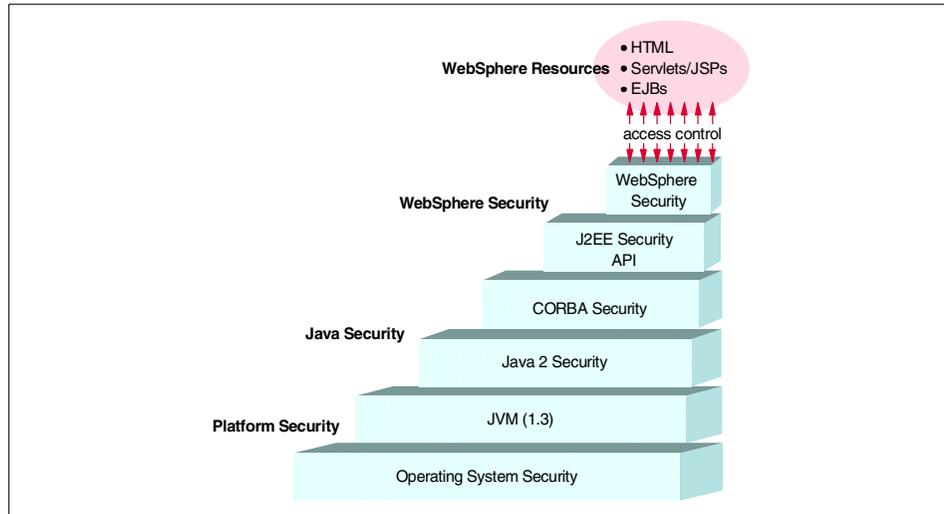


Figure 4-16 WebSphere security layers

4.4 Performance considerations

This section briefly discusses some performance considerations to keep in mind when using WebSphere security.

First, enabling WebSphere security has a performance cost of 10% to 20%. This performance hit can be eliminated by disabling security, of course. However, it may be more appropriate to consider other alternatives, such as:

- ▶ Disabling security on selected application servers.
- ▶ Using an HTTPS transport between the Web server and application server without enabling application server security (see “Configuring SSL between Web server and WebSphere Application Server” on page 270).

Another option is to tune the Security Cache Timeout, set using the WebSphere Security Center, as shown in Figure 4-9 on page 53. Security information related to EJBs, permissions, and credentials is cached. When the cache timeout expires, all cached information becomes invalid. Subsequent requests for the information result in a database lookup. Sometimes, acquiring the information requires invoking an LDAP-bind or native authentication, both of which are relatively costly operations in terms of performance.

In a simple 20 minute test run, raising the Security Cache Timeout from the default of 600 seconds to 6000 seconds avoided a timeout and resulted in a 40% performance improvement. Of course, the performance impact of tuning this property will depend on the specifics of your application, such as average session duration.

Finally, the SSLV3Timeout property specifies the time interval after which SSL sessions are renegotiated. The default of 9600 seconds is relatively high, so it should not impact performance, but should be reviewed in light of the specific application with which you are working.

The Secure Association Service (SAS) feature establishes an SSL connection only if it goes out of the ORB (to another ORB). Therefore, if all the beans are co-located within an ORB, then the SSL used by SAS is not expected to hinder performance.

Modify the SSLV3Timeout and other SAS properties by editing the `sas.server.props` and `sas.client.props` files. The files are located in the `<WebSphere install directory>\properties` directory.

4.5 Other security features of WebSphere

There are several other security features within IBM WebSphere Application Server V4, including the encoding of passwords and security interoperability with z/OS.

4.5.1 Encoded passwords

WebSphere stores passwords for:

- ▶ Accessing the administration repository
- ▶ The Administration ID to access the Administrator's Console
- ▶ Accessing key stores and trust stores

These passwords are stored in an encoded form. This is not a secure way of storing passwords in itself, but it does hide the passwords from a casual user who might get access to the system. The files which contain these passwords should still be secured from unauthorized access using the relevant file system policies.

If the passwords in these files need to be changed for any reason, then simply replace the hashed version (including the encoding tag, {XOR}) with plain text. When the WebSphere Administrative Server is restarted, the passwords will be rehashed.

The following table lists the files that contain passwords:

Table 4-8 Administration files which contain passwords

File name	Password types	File location
admin.config	Admin repository password	C:\WebSphere\AppServer\bin
sas.server.props sas.server.props.future sas.client.props (1)	WebSphere Administrator Key and trust store passwords	C:\WebSphere\AppServer\properties
initial_ssl.properties (2)	Key and trust store passwords	C:\WebSphere\AppServer\properties

Note: The key and trust store passwords in the sas.client.props are not encoded.

There is a tool for encoding the passwords in properties files, especially in SAS properties files. The tool is *<WebSphere install directory>\bin\PropFilePasswordEncoder.bat*. Change your directory where the SAS properties file is located, then use the following command to encode the passwords in the sas.client.props file:

```
PropFilePasswordEncoder sas.client.props -SAS
```

Note: The passwords in the *initial_ssl.properties* file are encoded, but if you replace them with plain text passwords, they will not be re-encoded. This is because this file is loaded only once when the admin service is started for the first time. Any changes to the SSL configuration should be made through the Administrator's Console and not through this file.

User names and passwords are also shown in XMLConfig exports of the configuration. Again, the passwords are exported in the encoded format.

4.5.2 Security interoperability with z/OS

IBM WebSphere Application Server Advanced Edition supports interoperability between application servers running on UNIX or NT platforms and application servers running on the z/OS platform. This support allows application servers on the UNIX or NT side to authenticate to the application server on the z/OS side and communicate securely. Unauthenticated requests from the UNIX- or NT-based application servers are rejected. Authentication is supported between application servers, not individual applications.

For more information about WebSphere and z/OS interoperability, refer to the WebSphere InfoCenter.

4.6 WebSphere Advanced Edition V4 ptf2

The ptf2 for WebSphere Advanced Edition V4 has been released; download it from the IBM WebSphere Web site at:

<http://www.ibm.com/software/websphere>.

It is highly recommended that you install this ptf level. It will also update the security features of WebSphere. The following list, distributed together with the ptf2 archive, shows the security changes in 4.0.2:

- ▶ PQ56055 - Spaces in LDAP user name problem
- ▶ PQ54156 - Administrative Console takes 10 minutes to come up when using LTPA
- ▶ PQ53688 - Trust file type does not correctly default
- ▶ PQ54789 - Potential authentication performance issues occur if user belongs to one or more groups with large memberships
- ▶ PQ55804 - InvalidTokenException results in authentication failure
- ▶ PQ51442 - sas.server.props should not be truncated
- ▶ PQ54124 - Authorization fails from unsecure to secure resource
- ▶ PQ56053 - LDAP does not accept special characters
- ▶ PQ56057 - Performance degradation LTPA token
- ▶ PQ51442 - Correct various security problems
- ▶ 110556.1 - Invalid Handling of invalid LTPA password on XML Import

- ▶ 110491 - The Remove action for dynamic properties does not work
- ▶ 100432 - Detailed exceptions during security initialization
- ▶ 112783 - isUserInRole() API throws an exception with JSP:forward tag
- ▶ 110911 - Trust Association does not work with WebSEAL 3.7.
- ▶ 116822 - Cannot enable crypto support at global level
- ▶ 110214 - When migrating from AEs to AE, a security realm name error occurs
- ▶ 110467 - Key File Name not required when using crypto card
- ▶ 110296 - Improve warning message when native code does not load



The sample used in this book

This chapter discusses the sample (Webbank) used in this book. The Webbank application has been introduced in the *WebSphere V4 Advanced Edition Handbook*, SG24-6176.

The original Webbank application has been extended with several security features, and with additional code.

The sample is provided as additional material together with this redbook; see Appendix A, “Additional material” on page 515 for details.

5.1 Sample application: Webbank

The sample used in this book, *Webbank*, is taken from the *WebSphere V4 Advanced Edition Handbook*, SG24-6176. The goal of this sample is to introduce all the security features documented in the book and also to provide examples and reusable code for programmers.

The following sections will go into the details of the sample application.

- ▶ First, the application structure will be shown.
- ▶ In order to reproduce the development environment for Webbank, we provide step-by-step guidelines on how to import the Webbank assets into the WebSphere Studio Application Developer (WSAD).
- ▶ The security roles are application-specific definitions. A whole section will be devoted to showing how to define these roles using the Application Assembly Tool (AAT) or WebSphere Studio Application Developer (WSAD).
- ▶ A very important part of this chapter deals with how to install (deploy) the Webbank application. We will run through the installation step by step.
- ▶ Security role mapping is related to the roles, described before the installation, but role mapping can be performed during or after the installation. There is an exception: role mapping can be defined from the WebSphere Studio Application Developer (WSAD).
- ▶ The Single-Sign On capabilities of WebSphere are exercised using a Lotus Domino example. This example requires a small Domino application, which will be described.
- ▶ At the end of this chapter, the extensions to the base Webbank application will be introduced. These extensions are described in later chapters of this book.

Two versions of the Webbank sample code are shipped with this redbook. One is *webbank6520_null.ear* and the other is *webbank6520.ear*. The first is the base code, without any security definitions or functional extensions. The second one includes all security features defined and all function extensions implemented.

Thus, anyone can start from the base code and enhance it following the steps described in this book. The fully developed code is always there for you to check or reuse some of the elements.

5.1.1 Base Webbank application structure

The following diagram shows the elements of the basic Webbank enterprise application.

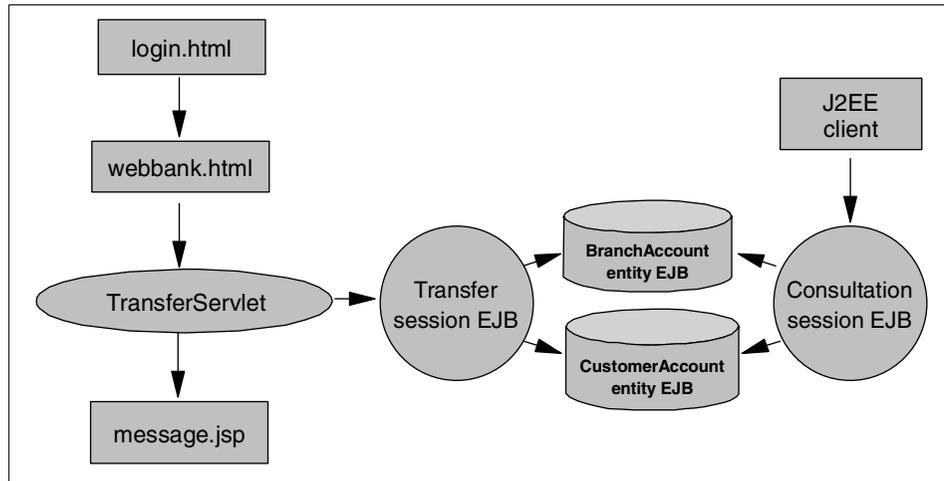


Figure 5-1 Site map for the Webbank application

The `login.html` page is the entry point for the application in case form-based login has been selected as the authentication mechanism. The sample code shipped with the redbook is set to use form-based login by default.

The `webbank.html` page is the first page the user sees once he or she is authenticated. It is a query page, where the user can set up a transaction from a branch account to a customer account or vice-versa. Figure 5-2 on page 76 shows an excerpt from `webbank.html`.



Figure 5-2 Webbank application: transfer page

The *TransferServlet* servlet controls the application flow and makes the connection between presentation and business logic. It initiates the transfer using a session Enterprise Java Bean (EJB) to access the data.

The *Transfer EJB* is a session EJB which handles the real transaction between the accounts. It plays an interface role for the two entity EJBs.

BranchAccount EJB and *CustomerAccount EJB* are two entity EJBs representing the account data. The account data is persisted in a database, and the entity EJBs are the connection to the database and to the accounts.

Consultation EJB is a session EJB which handles queries of the accounts; it basically calls the Read methods of the entity beans.

J2EE client is a stand-alone Java application running on the client. It uses a session Enterprise Java Bean (EJB) to access the data and retrieve the required information.

Once the transaction has been performed, whether or not it was successful, a result page will be shown to the end user. The result page is generated by the *message.jsp*.

5.2 Importing the sample into WebSphere Studio Application Developer

The sample code distributed with this book is comprised of several pieces. The sample was developed using WebSphere Application Studio Developer (WSAD) V4.02. If you want to perform the development within this IDE, the following steps will guide you through importing all the assets into the development environment.

The following steps will show you how to import the fully developed Webbank sample.

1. The best way to handle a new project is to create a new directory for the whole project, for example: `c:\projects\webbank`.
2. Start WebSphere Studio Application Developer (WSAD) with the following parameter: **-data c:\projects\webbank**.

```
wsappdev.exe -data c:\projects\webbank
```

You can also create a shortcut for WebSphere Studio Application Developer with this parameter on the desktop.

WebSphere Studio Application Developer will start in a minute with a new, empty project.

Import the Webbank enterprise application project:

3. Open the **J2EE perspective** if it is not already open.
4. Select **File -> Import** from the menu, then select the **EAR file** from the list. Click **Next**.
5. In the next window, browse for the **webbank6520.ear** file in the *EAR File* text-box. Type in the Enterprise Application project name: `webbank`. Click **Next**.
6. In the next window, check that every module has every dependent module selected, then click **Next**.
7. In the *EAR Modules* window, click **Finish**.
8. There will be errors after importing the EAR file, because of the missing Java libraries. Switch to the **Navigator view**, open the **.classpath** file under the `WebbankWeb` folder, and add the following lines to the end of the file (prior to the `<classpathentry kind="output"... />` tag). There are nine (9) new classpath variables, so make sure that you do not mistype them, because if the XML structure is invalid and you save the file, the project will crash.

```
<classpathentry kind="var" path="SOAPJAR"/>
<classpathentry kind="var" path="SOAPSECJAR"/>
<classpathentry kind="var" path="XERCESJAR"/>
```

```

<classpathentry kind="var"
path="SERVERJDK_PLUGINDIR/jre/lib/ext/ibmjceprovider.jar"/>
<classpathentry kind="var"
path="SERVERJDK_PLUGINDIR/jre/lib/ext/ibmjsse.jar"/>
<classpathentry kind="var" path="WAS_PLUGINDIR/lib/ejbcontainer.jar"/>
<classpathentry kind="var" path="WAS_PLUGINDIR/lib/ujc.jar"/>
<classpathentry kind="var" path="WAS_PLUGINDIR/lib/iwsorb.jar"/>
<classpathentry kind="var" path="WAS_PLUGINDIR/lib/security.jar"/>

```

You may also have to add the following classpath variables. Open the preferences window from the menu by selecting **Window -> Preferences**. Select the node by clicking **Java -> Classpath Variables**, then add the following entries (replace the `<WSAD install path>` with your path):

```

SOAPJAR <WSAD install
path>/plugins/com.ibm.etools.webservice/runtime/soap.jar
SOAPSECJAR <WSAD install
path>/plugins/com.ibm.etools.websphere.runtime/lib/soap-sec.jar
XERCESJAR <WSAD install path>/plugins/org.apache.xerces/xerces.jar

```

Click **OK** to close the Preferences window.

Import the CustomRegistry Java project:

9. Open the Java perspective, and create a new Java project called CustomRegistry.
10. Select **File -> Import** from the menu; the Import wizard appears, select the **Zip file** from the list, then click **Next**.
11. Browse for the **CustomRegistry.zip** file from the Zip file text-box; select the **CustomRegistry** folder from the *Folder* text-box, then click **Finish**.
12. During the import, a message box will ask about overwriting the .classpath file; select **Yes**.

Import the WebbankClientApp Java project just as you did for the CustomRegistry Java project:

13. Create a new Java project called WebbankClientApp.
14. Start the Import wizard, then import WebbankClientApp.zip into the WebbankClientApp folder; do not forget to overwrite the .classpath file when prompted.

Import the WebbankWebserviceClient Java project just as you did previously:

15. Create a new Java project called WebbankWebserviceClient.
16. Start the Import wizard, then import the WebbankWebserviceClient.zip into the WebbankWebserviceClient folder; do not forget to overwrite the .classpath file when prompted.

Import the Domino sample:

17. Create a new simple project, with the name WebbankDomino.
18. Start the Import wizard, then import the WebbankDomino.zip into the WebbankDomino folder. It will import two files: *Webbank.ntf* and a new transfer page for the Webbank application: *webbank.html*.

Create the server for the WebSphere Test Environment:

19. Open the Server perspective in WebSphere Studio Application Developer.
20. Select **File -> New -> Server Instance and Configuration** from the menu; the Server Configuration wizard is started.
21. In the first window, type in the server name: Webbank server, and specify the Folder in the text-box: Servers; then select the Server instance type: **WebSphere Servers -> WebSphere V4.0 Test Environment** (see the result in Figure 5-3). Click **Next**.

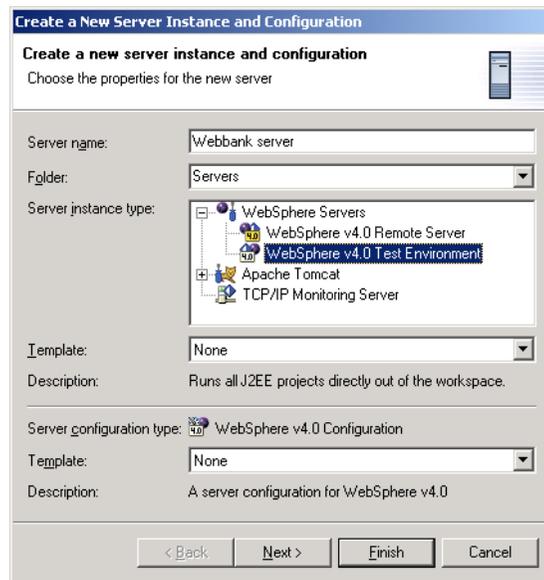


Figure 5-3 Creating a new server instance and configuration

22. A message box will prompt you: Do you want to create a new server project with the name Servers? Click **Yes**.
23. The next window asks for the HTTP port number for the test server; the default value is 8080. You can leave the port number at **8080**, or if you are not using the machine only for development, then it is more convenient to set the port number to **80**.

24. Click **Finish**. The new server instance and configuration will be generated. You can find more information about server instances and server configurations in the WebSphere Studio Application Developer help resources.
25. You now need to add the webbank project to the server configuration. In the lower left-hand corner is the *Server configuration* window. Right-click **Server Configurations -> Webbank server** and select **Add Project -> webbank**. You will get the following result:

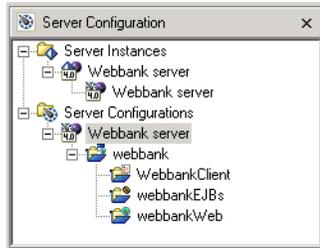


Figure 5-4 Server configuration for Webbank

26. You will next define the data source for the sample application. Double-click **Server Configurations -> Webbank server** under the *Server Configuration* window. An editor appears for the server configuration.
27. Switch to the **Data source** tab, then select the **Db2JdbcDriver** from the JDBC driver list.
28. Click **Add** next to the *Data source defined in the JDBC driver selected above* listbox.
29. Define the datasource by specifying the following (these are case sensitive):
- Name: webbank
 - JNDI name: jdbc/webbank
 - Database name: webbank
 - Default User ID: webbank
 - Default user password: webbank
30. Save and close the configuration file.

After these steps are performed, the development and test environment is set for the Webbank sample application. If everything was correct in the setup and configuration, only one warning should appear in the task list:

- ▶ Description: /webbank/j_security_check - Broken link.
Resource: login.html.
- ▶ Description: /webbank/ibm_security_logout - Broken link.
Resource: webbank.html

5.3 Defining security roles

In this section, we will define five security roles for the Webbank application:

- ▶ **Manager**
Only users in this group will be allowed to transfer more than \$5000 in a single transaction. This check is done programmatically within the Webbank application.
- ▶ **Employee**
Employees can transfer funds between customer and branch accounts, and can view account balance information. WebSphere security handles access to the resources to which users in this role have access.
- ▶ **Consultant**
Consultants have access to the Webbank application and can provide consulting services to the customers, but they cannot perform a transfer.
- ▶ **AllAuthenticated**
All authenticated users can view the balance of an account. This role uses a *special subject* which is predefined in WebSphere.
- ▶ **Everyone**
This is a special role used to allow access to resources when no security challenge has been put forth. We will define this role to the Webbank application for demonstration purposes, but it will not be used.
- ▶ **DenyAllRole** (this will be automatically defined during installation, unless you want to predefine it)

Once you have protected one of an EJB's methods, then all of the others are automatically protected. By assigning a role to that method, you allow the users associated with that role access to the method. No one will have access to any other methods that do not have a role assigned to them. WebSphere does this by using `DenyAllRole`. This is a special role which is automatically defined and can be assigned to resource methods which have not been protected. No users or groups are associated with this role.

There are two *special subjects* defined in WebSphere and the Application Assembly Tool: *All authenticated users* and *Everyone*. These special subjects are a quick way to configure a set of users to a security role without having to individually select users and groups.

The *All authenticated users* subject grants all users defined in your user registry access to the resources associated with the role. The *Everyone* subject grants all users access to the role whether or not they are in your registry, that is, they do not need to be authenticated to be given access to the resources associated with the role.

Security roles for an enterprise application can be defined in three places:

- ▶ At the Enterprise application level
- ▶ In an EJB module
- ▶ In a Web module.

A security role can be defined in any of these places and will be available in all of the other places too: if you define a security role to the WAR file, for example, it is also configured to the EAR and the JAR. However, if you want to assign particular users to the role, then you must do this at the Enterprise application (EAR) level.

5.3.1 Setting up users and groups in LDAP

This section will provide information about the users and groups you can register in a directory service for the sample application. This section will not document how to set up these users and groups; for more information about registering new items for a directory service, look for the native help or administration guide of the directory server.

Create the following users:

- ▶ Bob
- ▶ John
- ▶ Steve
- ▶ Andrew

Do not forget to set the password and the user ID for each user.

Create the following groups with the users included:

- ▶ Employee: Bob
- ▶ Manager: John
- ▶ Consultant: Steve

These users will have different privileges in the sample application; try the application with all users.

5.3.2 Security roles with Application Assembly Tool

We will first create a new security role for our Web module called *Manager*. Load the Webbank archive into the AAT.

1. Expand **Webbank** -> **Web modules** -> **webbankWeb**, and select **Security Roles**.

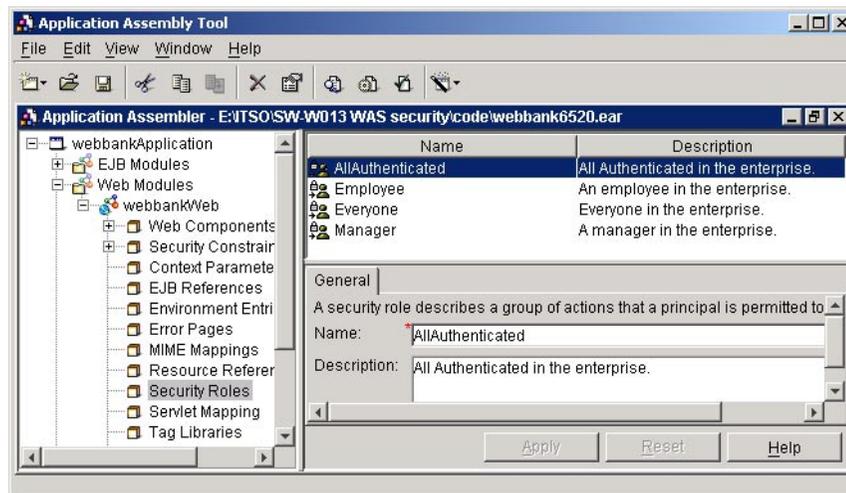


Figure 5-5 Web module security roles window

2. Right-click **Security Roles** and select **New**. You will be presented with the New Security Role window.



Figure 5-6 New Security Role window

3. Enter the Name: Manager and the Description: A manager in the enterprise. Click **OK**. You will now see your new security role displayed in the list in the right-hand pane.

Now expand **EJB Modules** -> **webbankEJBs** and select **Security Roles**. You will see the new Manager role in the list here as well.

Now define the role *Employee* and *Consultant*. Use the instructions above with the following field values:

Table 5-1 Field values for Employee and DenyAllRole roles

Name	Description
Employee	An employee in the enterprise
Consultant	A consultant in the enterprise

Note: It is possible to assign individual users and groups to these roles through the AAT. However, because the AAT does not interface with your user registry, it is up to you to make sure that you enter the names exactly as they are in the registry. Obviously, there is a risk of typographical errors. This is why we have chosen to leave this until the application is deployed into WebSphere. WebSphere can search the registry and you can choose the users and groups from a list.

Now we will define the last two roles, *AllAuthenticated* and *Everyone*. For these roles, we will also assign a special subject.

1. In the AAT, collapse all sections, leaving just the Webbank application expanded.

Note: We are defining these roles at the Enterprise application level, rather than within a Web module or EJB module, because we are going to assign users to these roles through the special subjects.

2. Right-click **Security Roles** and select **New**. You will be presented with the New Security Role window.

Note: This time, you will see that there are two tabs in this panel: General and Bindings.

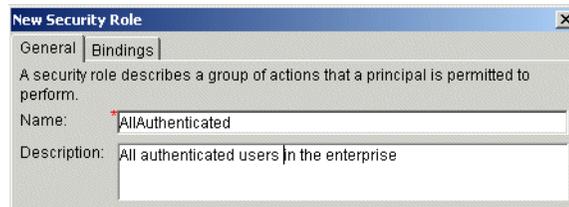


Figure 5-7 Enterprise application New Security Role window

3. Enter the Name: AllAuthenticated and the Description: All authenticated users in the enterprise. Switch to the **Bindings** tab.
4. Next to the Special subjects pane, click **Add**. You will be presented with the Add Special Subjects window.

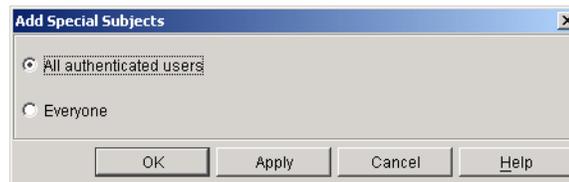


Figure 5-8 Special subjects for role binding

5. Select **All authenticated users** and click **OK**.
6. Click **OK** in the new Security Role window. You will now see your new security role displayed in the list in the right-hand pane.

Now define the role Everyone. Use the instructions above with the following field values:

Table 5-2 Field values for Everyone role

Field name	Values
Name	Everyone
Description	Everyone in the enterprise
Special subject	Everyone

5.3.3 Security roles with WebSphere Studio Application Developer

We will now show how you can configure the security roles from WebSphere Studio Application Developer (WSAD). Using WebSphere Studio Application Developer rather than the AAT allows your deployers to use the same tool and team environment as the developers and Web designers in the same project.

The following instructions have results similar to those achieved using the instructions for the AAT. The key differences are that you cannot assign special subjects to roles and that roles defined to one module are not automatically visible to the others.

1. Start WebSphere Studio Application Developer with the Webbank project.
2. Switch to the **J2EE** perspective or open this perspective if it is not open already.
3. Select the **Navigator** view and expand the **webbankWeb** folder, then the **webApplication** and the **WEB-INF** folders.
4. Double-click the **web.xml** file. On the right-hand side, you will see the Web application deployment descriptor configuration pane. Select the **Security** tab.

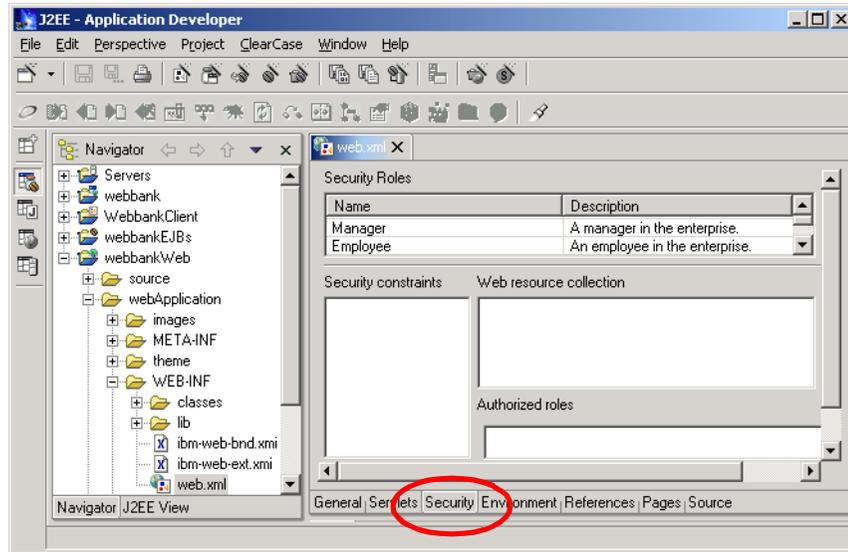


Figure 5-9 Security pane for application deployment descriptor

- In the right-hand pane, next to Security Roles, click **Add**. You will see (New Security Role) appear in the Name column of the Security Roles section.

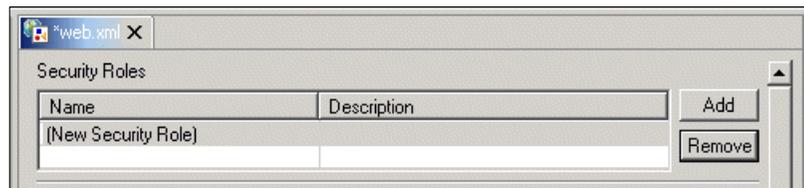


Figure 5-10 Adding a new security role

- Select this new role and change the name to **Manager**. Add a description of A manager in the enterprise.
- Now repeat steps 5 and 6 and add the roles **Everyone**, **Employee**, **Consultant** and **AllAuthenticated**. Use the following field values.

Table 5-3 Roles

Role name	Role description
AllAuthenticated	All authenticated users in the enterprise
Everyone	Every user
Employee	An employee in the enterprise
Consultant	A consultant in the enterprise

- Save the changes you have made to the Web module deployment descriptor by pressing the keys **Ctrl** and **S** simultaneously, then close the file.

Note: Unlike in the Application Assembly Tool, you must o add the roles to the EJB module and to the Enterprise application, it will not happen automatically.

- Now expand the **webbankEJBs** folder, then the **ejbModule** and **META-INF** folders. Double-click **ejb-jar.xml** to load the EJB module deployment descriptor configuration window. If you now select the **Security** tag, you will see that none of the roles you defined to the Web module are visible from here.
- Create roles called *AllAuthenticated*, *Everyone*, *Employee*, *Manager* and *Consultant* as before, but this time for the EJB module. You will notice that the process is slightly different, as you are prompted with an window requesting the Name and Description.



Figure 5-11 EJB security roles entry window

- Click **OK** and you will see the new role in the EJB descriptor window on the right-hand side. Save the new settings, then close the file.
- In the left-hand panel, expand the **Webbank** folder, then the **META-INF** folder. Double-click the **application.xml** file to load the enterprise application descriptor into the right-hand panel. Select the **Security** tab.
- Notice again that none of the roles is visible from here. Click **Gather Roles from Modules**. This will look for all the roles that have been defined in any Web or EJB modules, and populate the Enterprise application window with

them. Notice that any roles with the same name are consolidated into a single role.

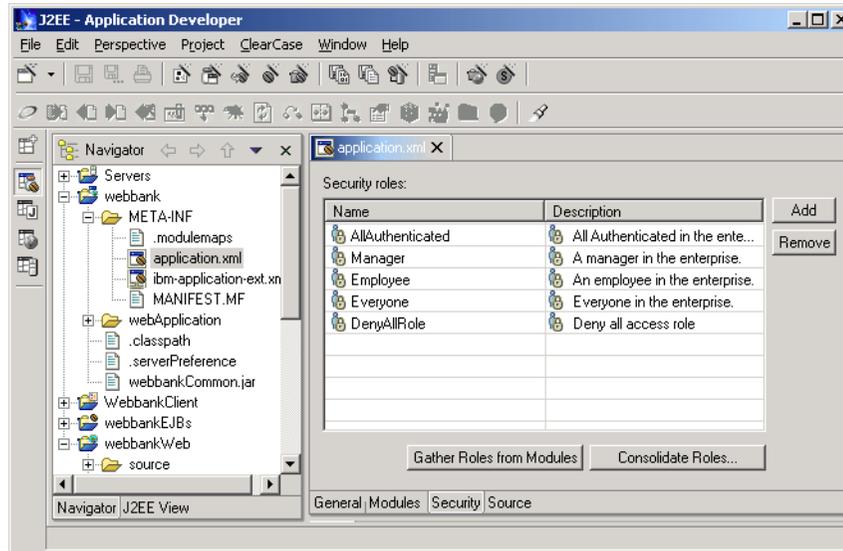


Figure 5-12 Populated Enterprise application security roles list

14. Save the settings by pressing the keys **Ctrl** and **S**, and close the file.

5.4 Installing the Webbank application

This section will describe how to install and deploy the Webbank application under WebSphere Application Server V4 Advanced Edition. We will look at:

- ▶ Creating the application server
- ▶ Setting up the data source
- ▶ Installing the enterprise application

For more information about the original Webbank application and application deployment, refer to the *WebSphere Advanced Edition V4 Handbook*, SG24-6176.

5.4.1 Creating the application server

The following steps will guide you through the process of creating an application server for the Webbank sample application.

1. Right-click **WebSphere Administrative Domain**, then click **Nodes -> <your server> -> Application Servers**, then select **New**.
2. The *Create Application Server* window appears. Under the General tab, type in the application server name: `Webbank server`, and change the module visibility to `Application`.
3. Click **OK**.
4. Wait until a message box appears saying: Command “`EJBServer.create`” completed successfully, then click **OK**.

5.4.2 Setting up the data source

The following steps will show you how to create the datasource for the Webbank application.

1. Create a user for the webbank database with the operating system.
The user we created for the sample was `webbank` with the password `webbank`.
2. We need to create the database and populate it with the sample data. Make sure that DB2 is running on your system, then open the DB2 Command window, or change to the DB2 user in UNIX.
3. Attach to the DB2 instance: **db2 attach to db2 user db2admin using db2admin**, where `db2` is the instance name and `db2admin` is the user name and the password; use your own settings for this command.
4. Create the database for the Webbank application: **db2 create database webbank**.
5. Connect, then modify the access permissions:
db2 connect to webbank
db2 grant connect, createtab on database to user webbank
db2 disconnect current
6. Connect with the webbank user this time: **db2 connect to webbank user webbank using webbank**.
7. Run the following script from the database directory to create the tables: **db2 -tf table.ddl**. The `table.ddl` was extracted from the EJB .jar file distributed within the enterprise archive.
8. This next script will populate the database: **db2 -tf insert.ddl**.

9. The database is ready now; enter **db2 disconnect current**.

The following steps will create the datasource under WebSphere Application Server.

1. Start the WebSphere Administrator's Console.
2. Right-click **WebSphere Administrative Domain**, then click **Resources -> JDBC Providers -> Sample DB Driver -> Data Sources**, then select **New**.
3. The *Data Source Properties* window appears. Fill out the required fields (the values are case sensitive):
Name: webbank
JNDI name: jdbc/webbank
Database name: webbank
User ID: webbank
Password: password
Confirm password: password
4. Click **OK**.
5. Wait until a message box appears saying: Command "Datasource.create" completed successfully, then click **OK**.

In this sample, we have just used the Sample DB Driver, which was already set up under WebSphere; in a production environment, you might want to set up your own database driver.

The user and password were chosen for this sample; make sure that the database user and the password for the datasource are kept secret.

5.4.3 Installing the enterprise application

In order to install the Webbank enterprise application, follow these next steps.

1. Select **Console -> Wizards -> Install Enterprise Application** from the menu. The Install Enterprise Application Wizard starts in a new window.
2. Leave the install application (*.ear) option selected, then select the **webbank6520.ear** file using the **Browse** button, then click **Next**.
3. A message may appear saying: This application contains method permissions. Do you wish to deny access to all unprotected methods? Click **Yes**.
4. The *Mapping Users to Roles* window appears. You will see all the roles listed here. You can do the mapping from here (refer to "Security role mapping

during installation” on page 94 for details), or you can change the settings made by the AAT or WebSphere Studio Application Developer. Click **Next**.

Note: If user mapping was done using WebSphere Studio Application Developer, some changes need to be performed. Unfortunately, there is a bug present: the special roles, *AllAuthenticatedUsers* and *Everyone*, are written improperly.

You should select these roles and do the mapping using the **Select...** button.

5. The *Mapping EJB RunAs Roles to Users* window appears next. You can perform the Run-As mapping for the application from here (refer to Section 7.4.4, “Run-As mapping during deployment” on page 152). Click **Next**.
6. The *Binding Enterprise Beans to JNDI Names* window appears next. The four EJBs should be already binded; click **Next**.
7. The *Mapping EJB References to Enterprise Beans* window appears next. The mappings should be already finished; click **Next**.
8. The *Mapping Resource References to Resources* window appears next. There is nothing to do here, so click **Next**.
9. The next window is titled *Specifying the Default Datasource for EJB Modules*. Select **webbankEJBs**, then click **Select Datasource**. Select the previously created **webbank** datasource, then click **OK**. Go to the next window by clicking **Next**.
10. The *Specifying Data Sources for Individual CMP Beans* window appears next. The data sources should be already assigned to the CMP beans, click **Next**.
11. The *Selecting Virtual Hosts for Web Modules* window comes next. The **webbankWeb** module is already mapped to the **default_host**; if you want to use a different host, use the **Select Virtual Host** button.
12. The next window is titled *Selecting Application Servers*. Two modules are listed here: **webbankEJBs** and **webbankWeb**. Select the two entries using the Shift key for multiple selection, then click **Select Server**. Select the **Webbank server** created previously, click **OK**, then click **Next** to get the next window.
13. In the final window, *Completing the Application Installation Wizard*, click **Finish**.
14. A final question appears: Regenerate code? Click **No**.
15. Wait until a message box appears saying: Command “EnterpriseApp.Install” completed successfully. Then click **OK**.

5.4.4 Starting the application

Follow these next steps to start the Webbank application.

1. Right-click **WebSphere Administrative Domain**, then click **Nodes -> <your server>**. Select **Regen Webserver Plugin**. After regeneration, it will take 60 seconds, by default, until the new settings are available.
2. Right-click **Webbank server** under the Application Servers folder (**WebSphere Administrative Domain -> Nodes -> <your server> -> Application Servers** folder), then select **Start**.
3. Wait until a message box appears saying: Command “Webbank server.start” completed successfully, then click **OK**.
4. Click **WebSphere Administrative Domain -> Enterprise Applications**, then right-click the **webbankApplication** server and select **Start**.
5. Wait until a message box appears saying: Command “Webbankapplication .start” completed successfully, then click **OK**.

5.5 Security role mapping

The defined Enterprise Application roles must be mapped to the user registry. This can be done with the Administrator’s Console using the Security Center or during the installation of the enterprise application.

5.5.1 Security role mapping with the Security Center

The following steps will guide you through the process of setting up role mappings using the Security Center.

1. Launch the Administrator’s Console, then select **Console -> Security Center** from the menu.
2. Switch to the **Role Mapping** tab, select the enterprise application (**webbankApplication**), then click **Edit Mappings....**
3. A new window appears; select the desired role, then click **Select**.
4. There are three options at this point.
 - a. You can select the special role **Everyone (no authentication)** for the role.
 - b. **All authenticated users(*)** is the other available option.
 - c. **Select users/groups** is the last option on the list.

Selecting either of the first two does not require further configuration; click **OK**, then continue the role mapping or close the Security Center.

Selecting the third option, as we do in our example, requires further steps:

5. Select the checkbox **Select Users/Groups** and search for users using a * (asterisk) in the Search field, then click **Search**.
6. A number of users and groups will appear in the list *Available Users/Groups*.
7. Assign the desired users and groups you want to add to the role by selecting them, then adding them (using the **Add>>** button) to the Selected Users/Groups list.
8. Confirm the settings in every window by clicking **OK**.
9. Close the Security Center.

5.5.2 Security role mapping during installation

Application deployers can also perform role mapping for the enterprise applications during installation with the Administrator's Console Enterprise Application Installation Wizard.

After the wizard is started, the first window will ask for the Enterprise archive (.ear) file. The next window is titled *Mapping Users to Roles*, it allows you to map users and groups to the roles (see Figure 5-13).

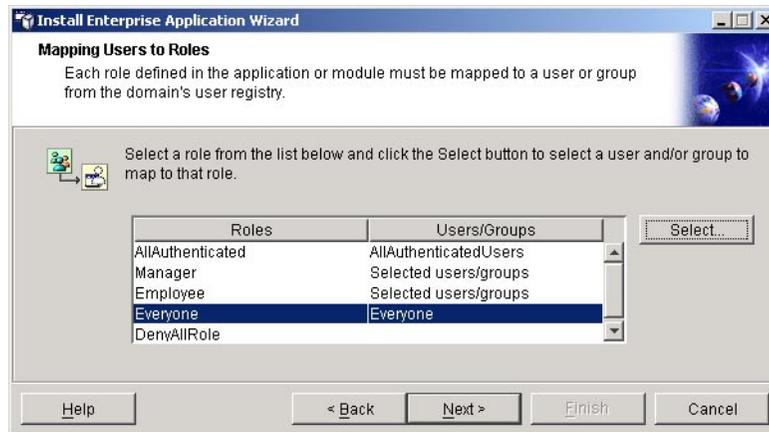


Figure 5-13 Role mapping during installation

The roles defined in the application are listed in the window. Select the role you want to map, then click the **Select** button. This will bring up the same selection window for mapping (see Figure 5-14).



Figure 5-14 Select Users/Groups window

Select the type of user or users you want to map to the role, then click **OK**.

After all the roles has been mapped, click **Next** in the installation wizard.

5.5.3 Security role mapping with the Application Assembly Tool

This section will show you how to use the Application Assembly Tool (AAT) to perform role mapping for an Enterprise application.

1. Start AAT and open the Webbank application.
2. Click the **Security Roles** node; the security roles appear in the upper right list.
3. Select the first role: **AllAuthenticated**, then switch to the **Bindings** tab below.
4. You will find three list-boxes for the mapping: *Groups*, *Users* and *Special subjects*. For the AllAuthenticated role, click **Add** next to the *Special subjects* list-box. A new window appears; select **AllAuthenticatedUser**, then click **OK**.
5. Click **Apply** at the bottom of the panel.
6. Select the next role: **Manager**; for this one, you must assign a group from the registry.
7. Switch to the **Bindings** tab again, then click **Add** next to the Groups list-box.
8. A new window appears, in which you must specify the Name for the group; type in Manager.
9. Click **Apply** at the bottom of the panel.

10. Perform the mapping for each role one after the other using the following table:

Table 5-4

Role	Bindings type	Bindings
Employee	Group	Employee
Everyone	Special subjects	Everyone
Consultant	Group	Consultant

11. Save the EAR file.

Note: Use the *common name (cn)* for the bindings instead of the fully *distinguished name (dn)*.

5.5.4 Security mapping with WebSphere Studio Application Developer

The following steps will guide you through the process of setting up security mappings using WebSphere Studio Application Developer.

1. Select the **J2EE** perspective in WebSphere Studio Application Developer, then switch to the **Navigator** view.
2. Click **webbank -> META-INF** folder, then double-click the file **ibm-application-ext.xmi**. The application extension editor should open.
3. Switch to the **Security Bindings** tab on the editor (see Figure 5-15).

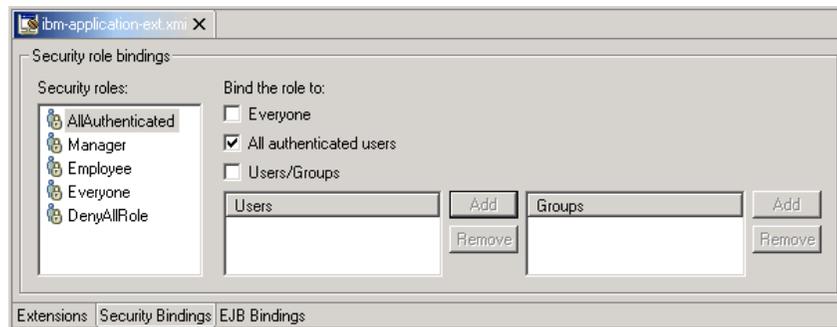


Figure 5-15 Application Extension Editor - Security bindings

4. Under *Security roles* are listed the roles from the application.

5. Select the roles one by one, then bind the roles to users or groups. The following options are available:

- Everyone
- All authenticated users
- Users/Groups

If **Users/Groups** is selected, the Users and the Groups window will be available. Developers and application assemblers can add users and groups clicking the **Add** button. The new entries are: (UserName) and (GroupName), by default. The entries can be edited by double-clicking them.

Bind the following users and groups to the roles from Table 5-5.

Table 5-5 Security bindings

Security role	Mapped to
AllAuthenticated	All authenticated users
Manager	Users/Groups, Group: Manager
Employee	Users/Groups, Group: Employee
Everyone	Everyone
DenyAllRole	(Do not assign anything)

Important: There is a problem with defining role mappings for an Enterprise application. Each role assignment has a wrong attribute in the <specialSubjects> tag. You must remove the `accessId=""` attribute from each tag. If you do not do this, WebSphere will not pick up the role mapping during the installation and you will have to do it manually again, either from the Security Center or during the installation.

Important: There is a bug in WebSphere Studio Application Server which may cause a problem when mapping special roles. The role *Everyone* is mistyped in the deployment descriptor as *everyone*. *AllAuthenticatedUsers* is also improperly typed using all lowercase: *allauthenticatedusers*. Correct these entries in *ibm-application-bnd.xmi* using the AAT or during installation, or edit the deployment descriptor directly. The correct values are:

- ▶ Everyone
- ▶ AllAuthenticatedUsers

5.6 Domino Webbank sample

To test the Single Sign-On (SSO) example included in this book, it is necessary to customize the following elements in the sample applications, by:

- Modifying the URLs included in the Webbank Comments Application template (webbank.ntf).
- Creating the Webbank Comment Application database from the template.
- Modifying the URLs included in the webbank.html page.

For more details about SSO, refer to Chapter 14, “Single Sign-On” on page 393.

Modifying the URLs included in the Webbank Comments Application template

The Webbank Comments Application database is a Lotus Domino application used in the example of SSO between WebSphere and Domino.

The application contains the URLs to connect to WebSphere from Domino. The purpose of this section is to explain how to deploy the Domino application in the Domino Server and how to modify these URLs to fit your environment.

To deploy the application, follow these next steps:

1. Copy the webbank.ntf template to the Domino Server data directory.
2. Start the Domino Administration Client, logged in as a user with administrative privileges, and open the Domino Server from the left server bookmark pane; click the **Files** tab.
3. Select **Templates only** in the Show me field to display all the templates located in the Domino data directory, as shown in Figure 5-16.

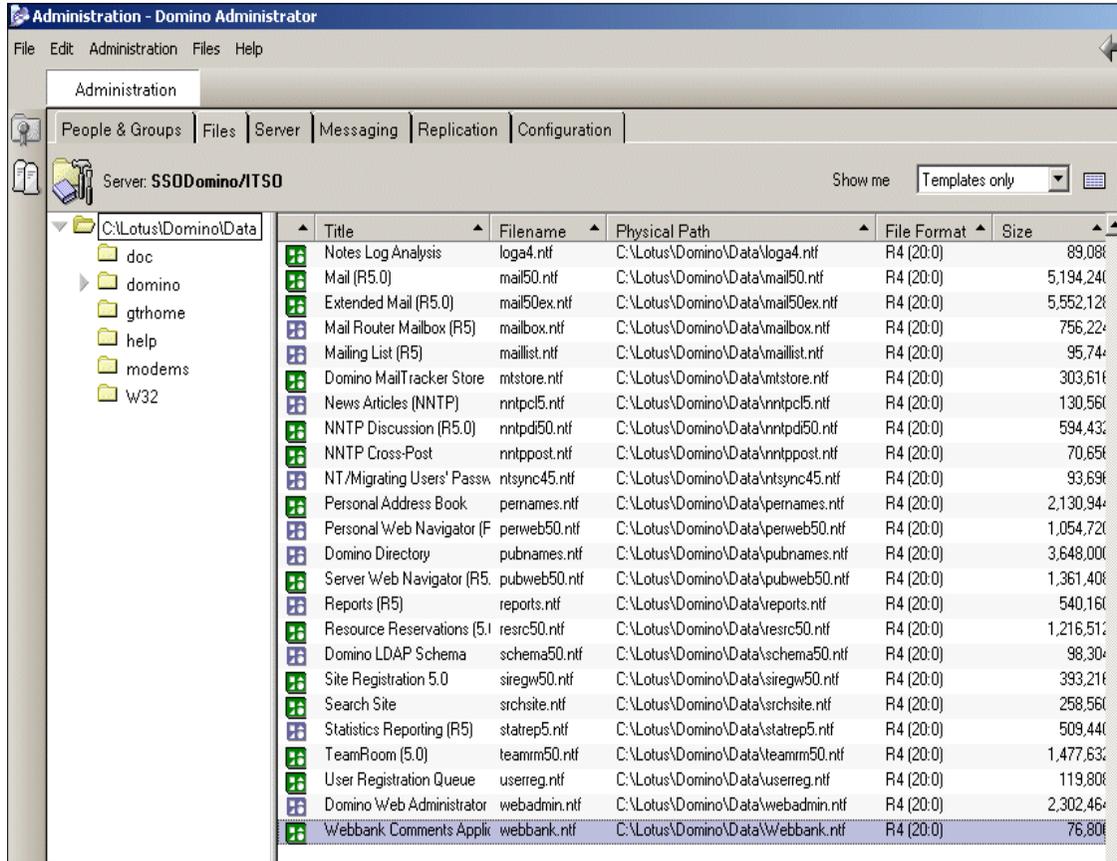


Figure 5-16 Selecting Webbank application template

4. Select the **Webbank Comment Application Template** and open the Database toolbar located on the right side of the tools pane, then select **Sign....**
5. A new Dialog box appears. Leave the default parameters selected and click the **OK** button (see Figure 5-17).

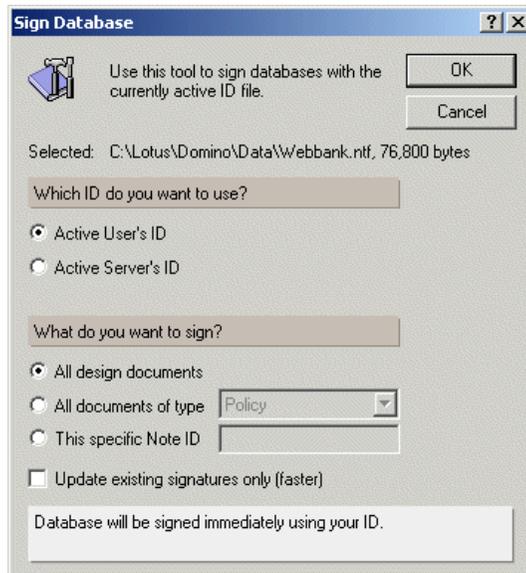


Figure 5-17 Sign Database dialog box

6. A new dialog box will appear stating "Your Name and Address Book does not contain a cross certificate for this organization" (for example: ITSO). Click **Yes** to create a new cross certificate.
7. All the design elements of the database will be signed with the actual ID. When the process is completed, a dialog box shows the number of databases processed and the number of errors that occurred (if any).

Also, you can check that the process has completed successfully by looking for the following message at the bottom of the Domino Administrator window:

Sign...Successfully processed SS0Domino/ITSO webbank.ntf

To modify the URLs included in the Comments form, follow the next steps:

1. Start Lotus Domino Designer and log in with the notes ID that you used before to sign the webbank.ntf database.
2. Select **File -> Database -> Open**. The new open database dialog box is displayed.
 - a. Select the server where the webbank.ntf template is located.
 - b. Select the **Webbank Comments Application template** from the drop-down database/templates list and click the **Open** button.
3. Click **Forms** in the Design window and then double-click the **Comments** form to open it, as shown in Figure 5-18.

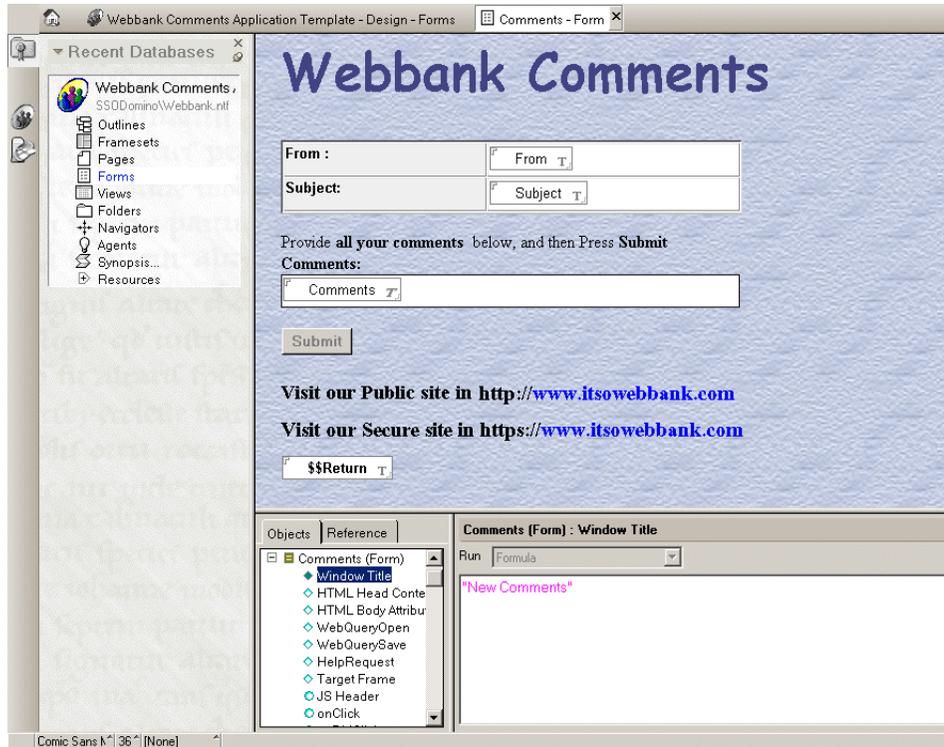


Figure 5-18 Webbank Comments form in Domino Designer

4. Place the cursor on top of the www.itsowebbank.com link and click the right mouse button. Point to **HotSpot Properties...**
5. A new HotSpot Properties dialog box appears. Select the **Content** section and type in the full URL address for your public WebSphere site in the Value field as `http://<your WebSphere server>/webbank`. Close the dialog box (see Figure 5-19).

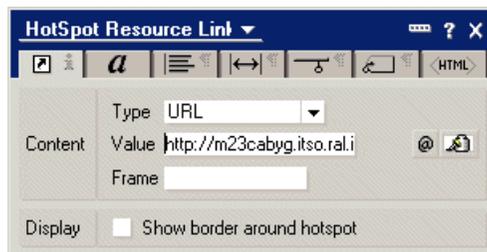


Figure 5-19 Changing the URL value

6. Repeat the same procedure to change the full address to your secure WebSphere URL: `https://<your WebSphere server>/webbank`.
7. Save the Comments form.

Creating the Webbank Comment Application database from the template

To create a new Webbank application database for use in the SSO example, follow these steps:

Start the Domino R5 Administration client with a notes administrator ID and then, from the Domino Administrator menu:

1. Choose **File -> Database -> New**. The new database Dialog Box is displayed.
2. Select the server where you want to create the new database.
3. Enter a title for the database, for example: Webbank Comments Application.
4. Enter a file name for the database, for example: webbank.NSF
5. Click the **Template Server...** button, select the Domino server that stores the Webbank Comments Application template (webbank.ntf) and highlight it.
6. Make sure that **Inherit future design changes** is selected, then click **OK**.

These settings are illustrated in Figure 5-20.

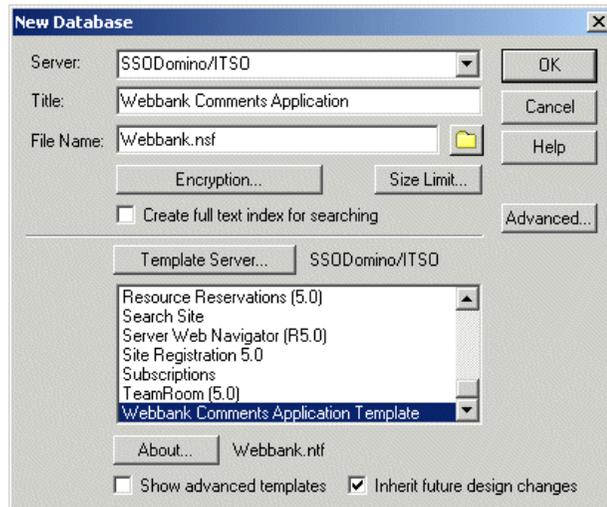


Figure 5-20 Creating a new webbank database from the template

Modifying the URLs in the webbank.html page

Once you have created the Webbank Comments Application Database, it will also be necessary to modify the URLs included in the webbank.html page to go from WebSphere to Domino with the URLs that fit your environment.

To do this, follow these steps:

1. Open the webbank.html with a text editor.
2. Locate the tag and change to the Domino URLs for your environment, as shown below.

```
<a href="http://was01.itso.ral.ibm.com/webbank.nsf/Comments?OpenForm">
Please, send your comments (public)</a></h4>
```

```
<a href="https://was01.itso.ral.ibm.com/webbank.nsf/Comments?OpenForm">
Please, send your comments (secure)</a></h4>
```

At this point, you are ready to test the SSO example. For more details, refer to Chapter 14, “Single Sign-On” on page 393.

5.7 Security samples

The following sections will summarize the function extensions to the base Webbank example.

Securing Web service

In Chapter 9, “Securing Web services” on page 171, a very simple Web service will be developed from one of the EJBs. This Web service will be secured using WebSphere’s Web service security features.

Securing J2EE clients

In Chapter 8, “Securing J2EE clients” on page 155, three different J2EE clients will be shown. They are:

- ▶ A *servlet*: the servlet authenticates itself on the server-side to the EJB container.
- ▶ A *Java thin client*: it connects to an EJB and authenticates itself on the client-side.
- ▶ The *J2EE client* has the full J2EE runtime environment and the developer does not need to deal with authentication, since the runtime environment will take over that task.

CustomRegistry sample

The custom registry sample is explained in a short section in Chapter 10, “Programmatic security” on page 201, where three different registry implementations will be introduced (see Section 10.3, “CustomRegistry SPI” on page 206). The following registries can be found there:

- ▶ File-based Custom Registry sample from the WebSphere InfoCenter
- ▶ Using DB2
- ▶ Using MQ Series



Securing Web components

In this chapter, we will discuss how to configure security for Web components such as servlets, JSPs and static resources.

Static resources can be secured by WebSphere Application Server or by the Web server. In “Static components served by a Web server” on page 106, we will show you how to configure authentication and authorization of static resources belonging to a Web server.

In ““WebSphere Web module security” on page 112,” we explain how to configure security for static and dynamic resources belonging to WebSphere. We will demonstrate how to do this using the Application Assembly Tool that comes with WebSphere, and using WebSphere Studio Application Developer.

6.1 Static components served by a Web server

WebSphere Application Server can only secure resources that it owns. Therefore, any static pages served by the Web server and that you want to secure will need to be protected by a means other than WebSphere.

Most Web servers are able to secure the files that they serve. For example, IBM HTTP Server can protect its own resources, using:

- ▶ HTTP basic authentication
- ▶ HTTP digest authentication
- ▶ Digital certificate authentication

In “How to secure HTTP basic authentication for IBM HTTP Server” on page 106, we provide an example of how to configure IBM HTTP Server to secure static content with HTTP basic authentication. In “Managing access to IBM HTTP Server using .htaccess” on page 111, we explain how access to these static resources can be managed using *.htaccess*.

For more information on how to configure security for IBM HTTP Server, see the product documentation.

It is also possible to use external security products to protect the Web server resources, such as IBM Tivoli Policy Director. See Section 13.2.2, “Using Tivoli Policy Director to protect static pages” on page 358 for more details.

6.1.1 How to secure HTTP basic authentication for IBM HTTP Server

In this section, we are going to enable security for all the static documents in the C:\IBM HTTP Server\htdocs directory. We will test this using `http://localhost`.

It is possible to configure HTTP basic authentication for IBM HTTP Server (IHS) using the following user registries:

- ▶ **Files:** group names, user names and encrypted passwords are stored in files.
- ▶ **Database:** DBM and DB type database files are supported.
- ▶ **LDAP:** users and groups are defined in an LDAP server, such as IBM SecureWay Directory Server. This can be the same LDAP server that WebSphere uses for its user registry.

For our example, we will use LDAP, IBM SecureWay Directory Server, to store the users' information. The following instructions assume that you already have an LDAP server set up and populated with users. See “SecureWay Directory Server” on page 460 for details on how to do this.

If your LDAP server is remote from your Web server, you will need to install the SecureWay client on the Web server machine. The client can be installed on its own from the IBM SecureWay Directory Server 3.2.1 download, available from <http://www-4.ibm.com/software/network/directory/downloads/>. Make sure that you have configured the client to connect to your remote LDAP server.

The following steps will show you how to enable LDAP authentication for IHS.

To enable HTTP basic authentication with LDAP, we need to perform the following steps:

1. Create a configuration file for the Web server to use to connect to the LDAP server, including the authentication type.
2. Add the LDAP module to the Web server configuration file.
3. Set the scope of the authentication:
 - a. Test that you can access the Web server. Make sure that the HTTP Server service is started, then from a Web browser, enter `http://localhost`. You should *not* be prompted for a user name and password.
 - b. Decide which user name and password with which you want the Web server to connect to the LDAP server. Now store an encoded version of the password in a stash file. From a command prompt, enter:
C:\IBM HTTP Server\ldapstash <password> C:\IBM HTTP Server\ldap.sth
 - c. Now create an LDAP configuration file for the Web server called `C:\IBM HTTP Server\conf\ldap.prop`. There is a sample file, `ldap.prop.sample`, in the same directory, which explains what each of the directives means. For basic authentication, the following entries are included.

Example 6-1 LDAP directives

```
ldap.realm=LDAP Realm
ldap.url=ldap://9/24/105/98/o=webbank (a)
ldap.transport=TCP
ldap.application.authType=Basic
ldap.application.DN=cn=Joanna Hodgson,c=US,o=webbank (b)
ldap.application.password.stashFile=ldap.sth
ldap.user.authType=Basic
ldap.group.name.filter=(&(cn=%v1)(|(objectclass=groupofnames)(objectclass=groupofuniquenames)))
ldap.group.memberAttributes=member uniquemember
ldap.idleConnection.timeout=600
ldap.waitToRetryConnection.interval=300
ldap.search.timeout=10
ldap.cache.timeout=600
```

where

- i. `ldap.URL` is of the form `ldap://<hostName>/<BaseDN>`
- ii. `ldap.application.DN` is the DN by which the Web server authenticates itself to the LDAP Server.
- d. Now add the LDAP module to the Web server configuration. From a Web browser, go to the IHS server configuration. Go to `http://localhost` then select **Configure server**. When prompted, enter your Web server administration ID and password.

Note: If you have not set up an administration ID, then from a command prompt, enter:

```
C:\IBM HTTP Server\htpasswd -c C:\IBM HTTP  
Server\conf\admin.passwd <adminName>
```

When prompted, enter your chosen password and verify it.

Stop and restart both Web server services to make the changes effective.

- e. Select **Basic Settings -> Module Sequence**. Make sure that the Scope is set to **<GLOBAL>**, then select **ibm_ldap (IBMModuleLDAP.dll)** and click **Add**.

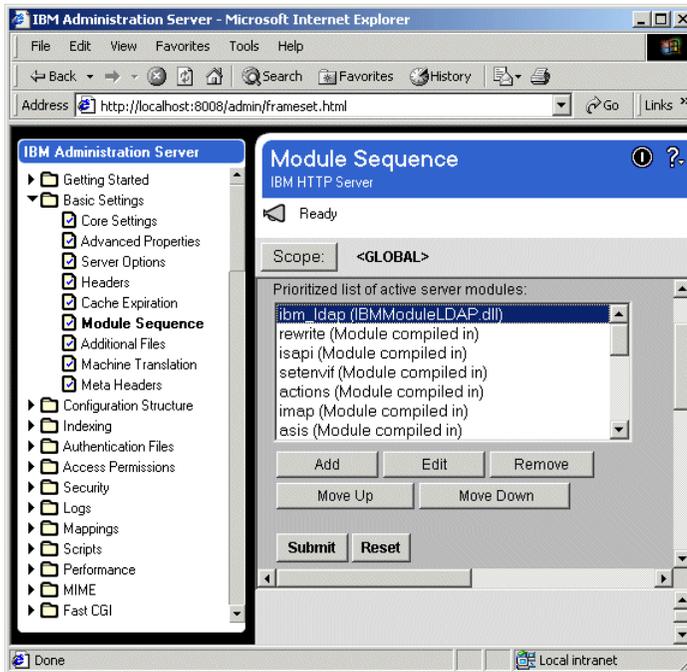


Figure 6-1 Configuring the LDAP module for IBM HTTP Server

- f. Click **Submit**.
- g. Next, set the scope and type of authentication. We urge you to secure all the documents in the C:\IBM HTTP Server\htdocs directory. From the Web server administration console, select **Access Permissions -> General Access**. Click the **Scope** button and select **<Directory C:\IBM HTTP Server\htdocs>**.



Figure 6-2 Choosing the authentication scope

- h. Now select **LDAP** as the authentication type and enter the name of the configuration file you created in step b above, `C:\IBM HTTP Server\conf\ldap.conf`. Enter an authentication realm name; we used `LDAP Realm`.

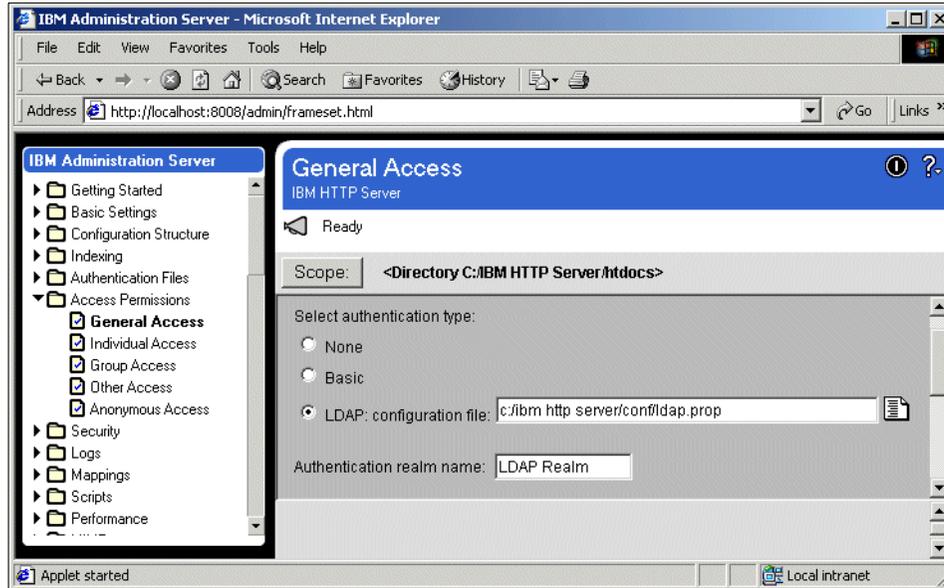


Figure 6-3 Finalizing the LDAP configuration

- i. Now click **Submit**.
- j. Restart the Web server, clicking the Restart Server button (see Figure 6-4).



Figure 6-4 Restart server button

- k. Now close the browser in which you were administering the Web server. Restart the browser and go to `http://localhost`. This time, you should be prompted for a user name and password. Enter any valid user in your LDAP registry.

Note: The configuration changes you made through the Web server administration tool added the following lines to the C:\IBM HTTP Server\conf\httpd.conf

```
LoadModule ibm_ldap_module modules/IBMModuleLDAP.dll
LDAPConfigFile "c:/ibm http server/conf/ldap.prop"
AuthName "LDAP Realm"
AuthType basic
Require valid-user
```

6.1.2 Managing access to IBM HTTP Server using .htaccess

By default, the Web server configuration is handled only by the Web administrator. The IHS configuration file, httpd.conf, explicitly enforces this with the following directive:

```
<Directory "C:/Program Files/Apache Group/Apache/htdocs">
    AllowOverride None
    Options None
</Directory>
```

There are occasions when this is a bit limiting. First, it means that all configuration changes require a restart of the server to be effective. Second, you might want to give an individual user or group of people the ability to configure their own area of a Web site. This is not possible with the default settings.

IBM HTTP Server can be configured to provide different levels of access on a per-directory basis, overriding directives in the server configuration defined in httpd.conf. This is done using configuration files called *.htaccess* within each directory over which you want to have this control.

When using these *.htaccess* files, note that for each request for a file, IHS will check to see whether there are any *.htaccess* files in the directory tree. These files can exist in multiple directories in the tree. In this case, the directives in the *.htaccess* file of the subdirectory take precedence over the directives in the parent directory.

Because IBM HTTP Server checks for the *.htaccess* files upon each request for a page, there is no need to restart the server for the configuration changes to take effect.

There are a number of directives that can be overridden. When dealing with security, we are interested in the *AuthConfig* category of directives. To override this category, change the directive in the httpd.conf file to:

```
<Directory "C:/Program Files/Apache Group/Apache/htdocs">
    AllowOverride AuthConfig
    Options None
</Directory>
```

For more information on how to use .htaccess see the Apache tutorial at <http://apache-server.com/tutorials/ATusing-htaccess.html>.

6.2 WebSphere Web module security

A Web module is made up of static resources (such as HTML files, images, sound files, etc.), servlets and JSPs. A description of how to configure security for each of these specific resource types can be found in the following sections. There are also some security settings applicable to all of these resources which are defined at the Web module level.

The Application Assembly Tool (AAT) is used to configure security for a Web module. Using it, you can configure:

► Security roles

The security roles for our Webbank application were defined in Section 5.3, “Defining security roles” on page 81. As noted there, security roles defined to a Web module are visible at the Enterprise application level; similarly, all roles defined to the application are also available to the Web module.

► Security constraints

A security constraint is a mapping between a security role and a set of resources. A resource can be a URL, in the case of static content, an entire servlet or JSP, or methods of a set of servlets or JSPs. There can be several security constraints for each Web module. If global security is enabled, then once a security constraint is set for a particular resource, it is secured.

► Authentication method

WebSphere supports three authentication methods:

– Basic authentication

The user name and password are encoded by the browser and included in the HTTP request. To secure the user information during transmission, this channel should be encrypted.

– **Client certificate authentication**

The client certificate is transported across an SSL encrypted channel to the Web server. The Web server then extracts the credentials from the certificate and forwards them to WebSphere along with the request.

– **Form-based authentication**

By default, the values that the end user supplies in the form are transmitted in clear text as parameter values in the HTTP request. To secure the user information during transmission, this channel should be encrypted.

The Application Assembly Tool (AAT) also has an option for *digest* authentication, but this option is not supported by WebSphere.

If a security constraint has been set but no authentication method for a Web module has been configured, the default is to use basic authentication.

The Webbank application will work with each of the authentication methods, as the Web module (`webbankWeb`) contains an HTML file for form-based authentication.

It is possible to configure each Web module, within an enterprise application, to use a different authentication method. This way, you could configure one Web module to use client certificates for your intranet users, for example, and another module to use basic authentication for your Internet users.

If you configure an authentication method default for all Web modules, then any Web modules that have not already been configured will inherit this setting.

The authentication method is configured from the AAT; the following steps will guide you through our example:

1. Load the Webbank application file into the AAT.
2. Expand **Webbank** -> **Web modules**, and select the **webbankWeb** module. In the right-hand pane, select the **Advanced** tab.

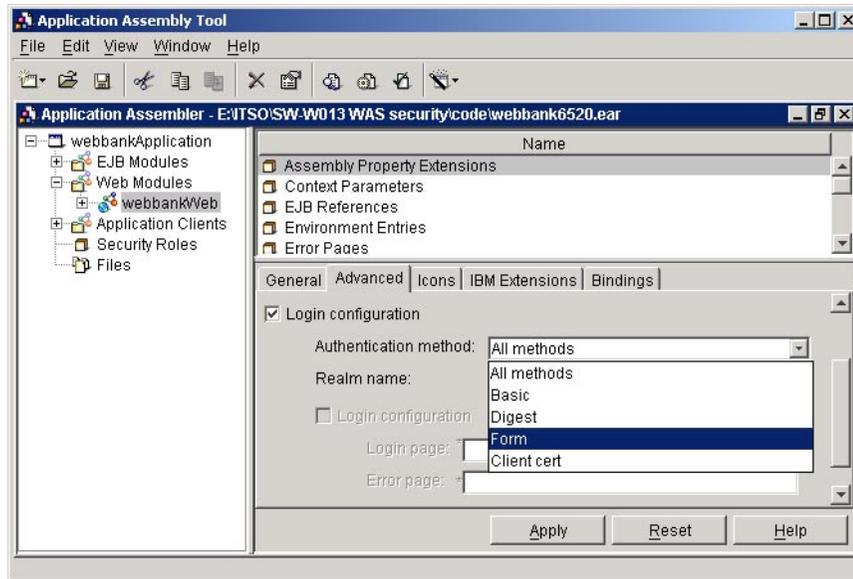


Figure 6-5 Authentication method pane

3. Select the **Login configuration** checkbox. This will enable the **Authentication method** drop-down menu. Now select the method you want to use : **Basic, Form** or **Client cert**.
4. Enter a Realm name; in our example we used **Webbank Realm**.

Definition: A *realm* is a security domain. Each application server resides within a realm.

5. If you selected **Form** for the authentication method, you now need to define the *Login* and *Error* pages that you will use for this.

Note: These files must be included in a Web module.

First, select the second **Login configuration** checkbox. Now enter the file names. For our Webbank application, enter:

- Login page: /Login.html
- Error page: /Login.html (in case the login failed, the user is redirected to the login page again)

Note: You must not include the context root for the Web module here, as it is assumed.

6. Click **Apply** for the changes to take effect.

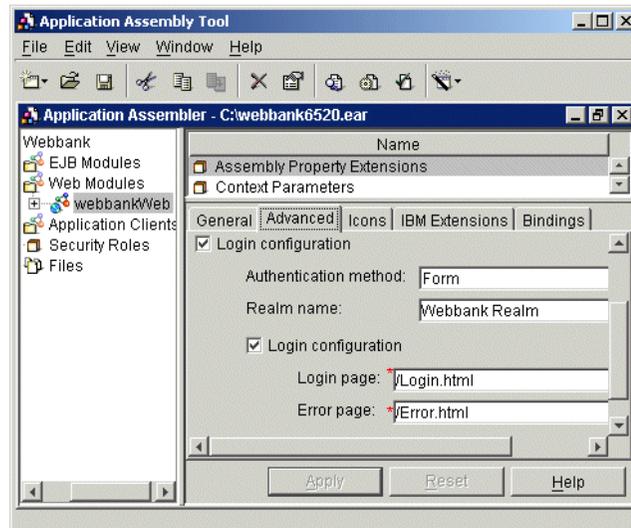


Figure 6-6 Completed Form login pane

You can now use these instructions to configure any other Web modules that you have in your enterprise application. If you have multiple Web modules for which you want the same authentication method, you can follow these instructions, but first select **Web Modules**, then the **Advanced** tab, and continue from step 2 above.

Below are the screen shots of the different types of authentication. To launch the application, point your browser to `http://<hostname>/webbank`.

Note: <hostname> should be the fully qualified host name of the WebSphere server, and not localhost or the IP address.

The basic authentication challenge is shown in Figure 6-7.



Figure 6-7 Basic authentication challenge

The form-based authentication challenge is shown in Figure 6-8.



Figure 6-8 Form-based authentication challenge

6.3 Securing the Web components

The following sections will describe the methods you can use to secure your Web components (static pages, servlets, JavaServerPages) with WebSphere Application Server.

6.3.1 Static pages served by WebSphere Application Server

WebSphere Application Server can secure static resources that it owns, that is, which reside on the Application Server and not on the Web server. According to the J2EE specification, these static resources must be packaged within a Web module (Web ARchive file). The static resources are then served up to the browser by the *File Serving Servlet* running inside the WebSphere Web container.

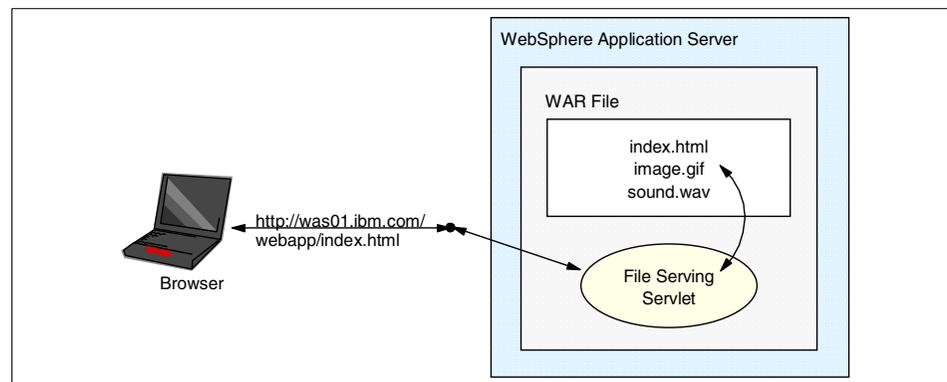


Figure 6-9 WebSphere serving a static file

For information on how to create a Web module to include static resources, see the WebSphere InfoCenter, section 6.3.1, “Assembling modules.” To ensure that file serving is enabled, see “Static pages served by WebSphere Application Server” on page 117.

In our Webbank sample application, we have several static resources, including several image files, the HTML files for the Form login, and the entry HTML page to our application (webbank.html).

Important: You must never secure the pages for the Form login, as you will be effectively securing a page with itself!

The following instructions show how to secure a static resource by setting a security constraint. Here we will configure webbank.html so that all authenticated users have access to it.

1. Load the Webbank archive file into the AAT.
2. Expand **Webbank -> Web modules -> webbankWeb** and select **Security Constraints**. Right-click **Security Constraints** and select **New**. You will be presented with the New Security Constraint entry window.
3. Enter the Security constraint name: `Webbank access`.
4. In the **Authorization Constraints** pane, next to Roles, click **Add**. You will be presented with a dialog box listing all the security roles that are defined for your application.
5. Select the **AllAuthenticated** role and click **OK**.

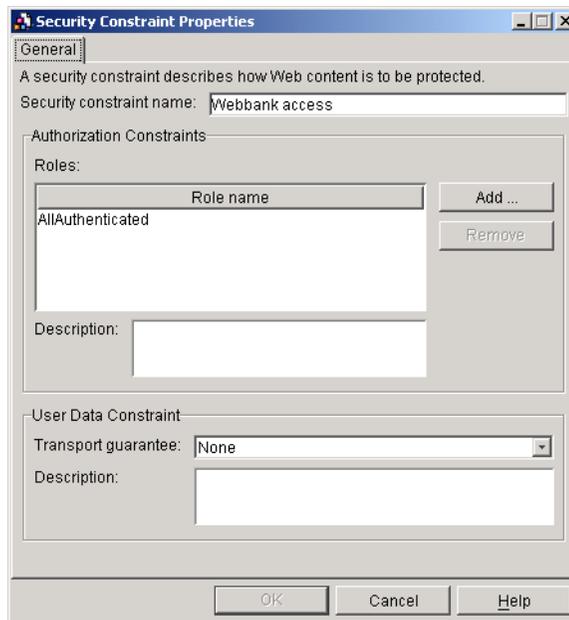


Figure 6-10 Completed New Security Constraint window

Note: The User Data Constraint section of this window allows you to configure a *Transport guarantee* which forces connections from a browser to come in only over an encrypted channel, that is, HTTPS. This only affects the connection between the browser and the Web server.

There are three options you can choose from:

▶ **None**

No constraint is applied. Requests from a browser can come in over HTTP or HTTPS.

▶ **Integral**

This ensures that data cannot be changed in transit. In practice, this means that a request must be transmitted over an SSL encrypted channel.

▶ **Confidential**

This ensures that data cannot be viewed in transit. In practice, this means that the request must be transmitted over an SSL encrypted channel.

See “Configuring the Web Server to support HTTPS” on page 239 for more information on configuring SSL.

6. Click **OK** to save the security constraint. You will now see your new security constraint (*Webbank access*) listed in the constraints window.
7. Expand the new constraint and select **Web Resource Collections**. Right-click **Web Resource Collections** and select **New**. You will be presented with the **New Web Resource Collection** entry window.
8. Enter the Web Resource Name: `Webbank access`.
9. Next to *URLs*, click **Add**. You will be presented with the Add URLs entry box.
10. Add the URL pattern `/` (forward slash on its own), then click **OK**. Now add a second URL pattern: `/webbank.html`.

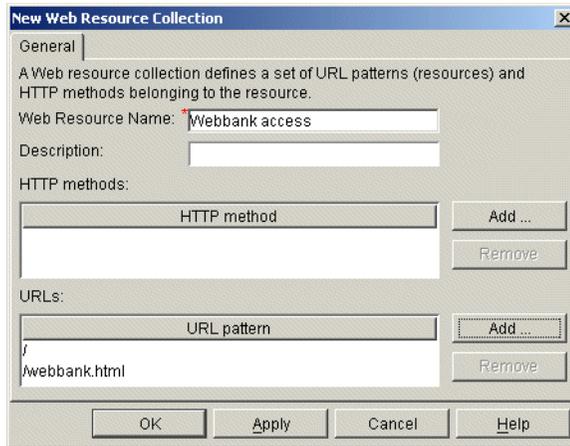


Figure 6-11 Completed Web Resource Collection window

11. Click **OK** to finish.

6.3.2 Servlets

WebSphere can secure its dynamic resources, such as servlets, using the method type. For example, the POST method of a servlet can be part of a different security constraint than the GET method. The full list of predefined methods that can be secured is:

- ▶ GET
- ▶ POST
- ▶ PUT
- ▶ DELETE
- ▶ HEAD
- ▶ OPTION
- ▶ TRACE

You might want to use this method if you had some dynamic content (HTTP_GET) that all authenticated Internet users can view and also had some administration function that you only wanted your staff to be able to perform (HTTP_POST). In this case, you would choose to define separate security constraints for the two methods.

If no methods are explicitly defined in the constraint, then all methods are secured.

For our Webbank example, we will configure one constraint for the *TransactionServlet* which defines both the GET and POST methods. We will assign this constraint to the *Employee* and *Manager* roles. First, load the Webbank application file into the AAT, then:

1. Expand **Webbank** -> **Web modules** -> **webbankWeb** and select **Security Constraints**. Right-click **Security Constraints** and select **New**. You will be presented with the **New Security Constraint** entry window.
2. Enter the Security constraint name: **Webbank Transfer**.
3. In the Authorization Constraints section pane, next to Roles, click **Add**. You will be presented with a dialog box listing all the security roles that are defined for your application. Select the **Employee** and the **Manager** roles and click **OK**.
4. Click **OK** to save the security constraint. You will now see your new security constraint (*Webbank Transfer*) listed in the Constraints pane.

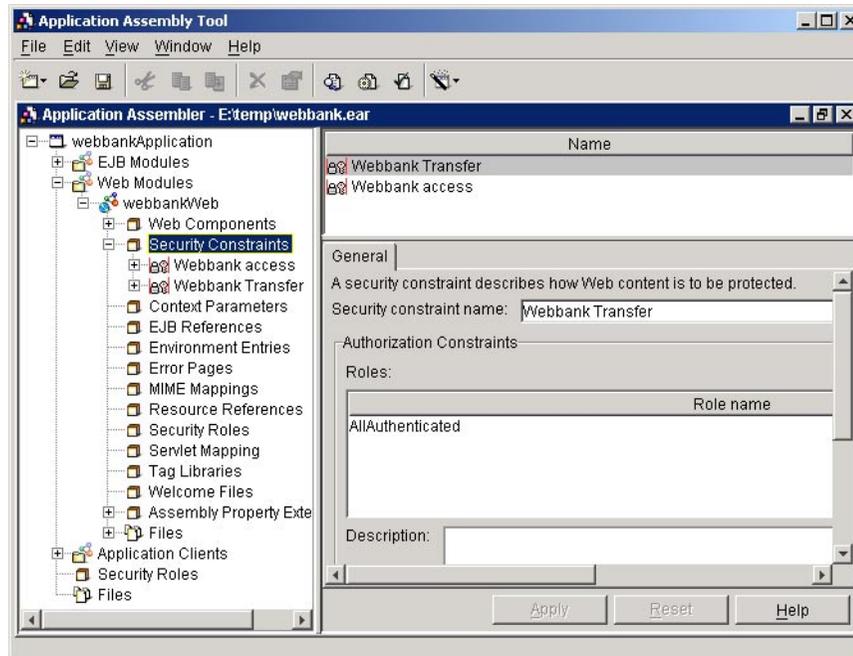


Figure 6-12 Security constraint for TransferServlet

5. In the left-hand pane, expand the new constraint and select **Web Resource Collections**. Right-click **Web Resource Collections** and select **New**. You will be presented with the **New Web Resource Collection** entry window.
6. Enter the Web Resource Name: **Webbank transfer**.

7. Next to *HTTP methods*, click **Add**. You will be presented with the Add HTTP methods entry box.
8. Add the HTTP method: GET then click **OK**. Now add a second HTTP method: POST.
9. Next to *URLs*, click **Add**. You will be presented with the Add URLs entry box. Add the URL pattern: /TransferServlet, then click **OK**.

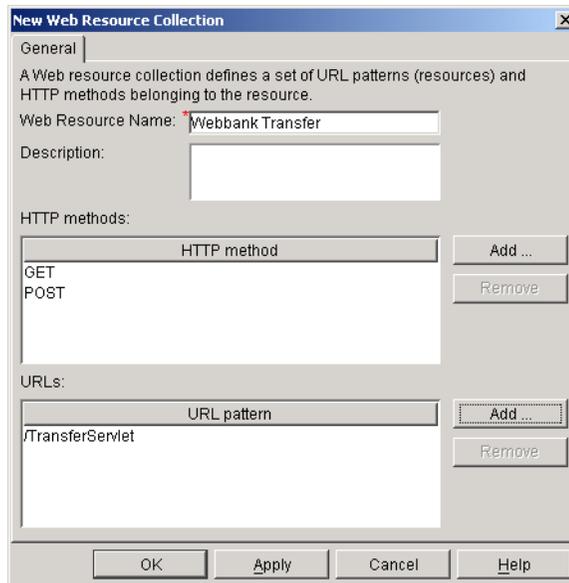


Figure 6-13 Completed Web resource collection window for TransferServlet

10. Click **OK** to finish.

Security role references for servlets

You can set up security role references for the servlets. These references help to separate the development from the deployment. Application developers can use any name for the roles, then, during deployment, the application deployers can make the link between the role name used in the source code and the role name used for the enterprise application.

Security role references can be defined using the Application Assembly Tool (AAT):

1. Start the Application Assembly Tool (AAT), then open the enterprise application archive, for example: webbank.
2. Select the node: **webbankApplication -> Web modules -> webbankWeb -> Web components -> TransferServlet -> Security Role References**.

3. Right-click the node, then select **New**.
4. A window pops up with the settings. Name is the name of the role used in the source code; type in `Employee`.
5. The Link is a selection list where you can select the roles defined for the Web module. Select **Employee**.
6. You can write a description for the entry if you need to.
7. Click **OK**.

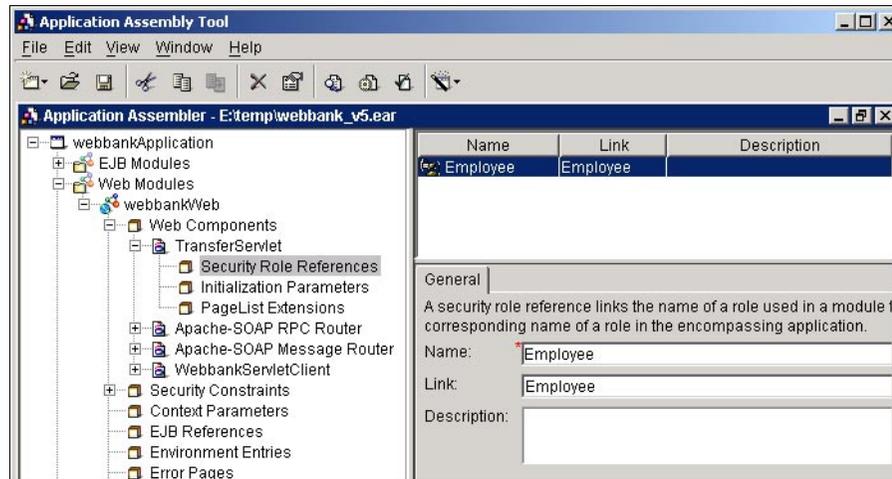


Figure 6-14 Security role references for servlets

You can also set security role references using the WebSphere Studio Application Developer (WSAD).

1. Start WebSphere Studio Application Developer with the Webbank project, switch to the J2EE perspective, select the **J2EE view**, then double-click the **Web modules -> webbankWeb** entry.
2. The web.xml editor comes up; switch to the Servlets tab.
3. There is a list-box called *Authorized roles*. Click **Edit** next to the list-box.
4. Select the **Employee** role.

This performs the reference a bit differently than the Application Assembly Tool (AAT); in this case, the name in the source code and the link to the application roles will automatically be the same: `Employee`. You can check the entry in the `web.xml` source by searching for the following tag:
`<security-role-ref>`.

5. Save and close the web.xml file.

Note: the security role references were not tested for servlets at the time this book was written.

6.3.3 JavaServer Pages (JSPs)

As for servlets, WebSphere can secure JSPs using the method type. JSPs which are defined as Web components in your Web module can be configured in exactly the same way as for servlets (see Section 6.3.2, “Servlets” on page 120).

In a classic Model-View-Controller application architecture, JSPs will return dynamic content as part of the presentation layer, not control the application flow or contain any business logic. The information presented by the JSP can only be accessed by first calling a servlet. In this scenario, it may not be necessary to protect your JSPs, even if the servlet is protected.

If the Model-View-Controller application architecture has not been followed or you want to secure your JSPs anyway, then take care not to secure any error pages that you would want to display to non-authenticated users.

The Webbank application has a single JSP used to report the result of a bank transfer. As this application conforms to the Model-View-Controller architecture, we will not apply any constraints to the JSP itself. Access to the Web application is controlled through TransferServlet, and no result will appear on the JSP accessing it directly.

6.4 Defining WebSphere Studio Application Developer security constraints

This section will show you how to define security constraints for the Web module using WebSphere Studio Application Developer. The security constraint settings are the same as in “Securing the Web components” on page 117.

1. Start WebSphere Studio Application Developer with the Webbank project.
2. Open or select the **J2EE perspective**, and switch to the J2EE view.
3. Open the Web Modules folder, then double-click the **webbankWeb** item.
4. Switch to the Security tab.
5. Define a new security constraint: click the **Add** button under the Security constraints list-box; a new item appears in the box, and one under the Web resource collection.

6. Select the new entry under the Web resource collection, then click **Edit** on the right side of the list-box; a new window appears.
7. In the *Web Resource Collections* window, select the **GET** and **PUT** methods under the HTTP Methods list. Then add the following URL pattern: `/TransferServlet`. Click **OK** to close the window.

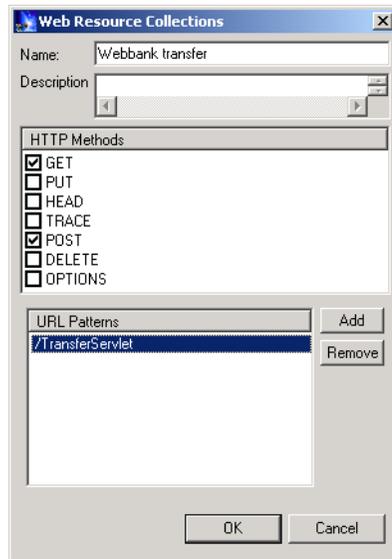


Figure 6-15 Web Resources Collection window

8. Now add the roles to this security constraint. Click **Edit** next to the *Authorized Roles* list-box; a new window appears with the defined roles.
9. Select the **Employee** and **Manager** roles. Fill out the Description field. This is tricky, as the security constraint name is stored in this description field. The Application Assembly Tool (AAT) can recognize the name and the description part. The security constraint name has to be the first string, and has to end with the following sequence: `:+:` (colon, plus sign, colon). So in the description field, type the following: `Webbank Transfer:+:`. Click **OK** to close the window.

Note: It is not necessary to fill out the description field, or to specify a name for the security constraint. Without specifying a name, the security constraint will appear in AAT as `<name not specified>`.

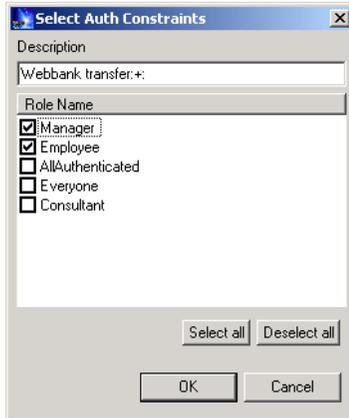


Figure 6-16 Select Auth Constraints window

10. Also set a security constraint for the webbank.html static page. Follow the previous steps 5 to 9, using the following values:

Security constraint name: Webbank access

Web resource collection name: Webbank access

Web resource collection - URL Patterns: / , /webbank.html

Web resource collection - HTTP Methods: GET

Authorized roles: AllAuthenticated

11. The result should look like that shown in Figure 6-17 on page 127.

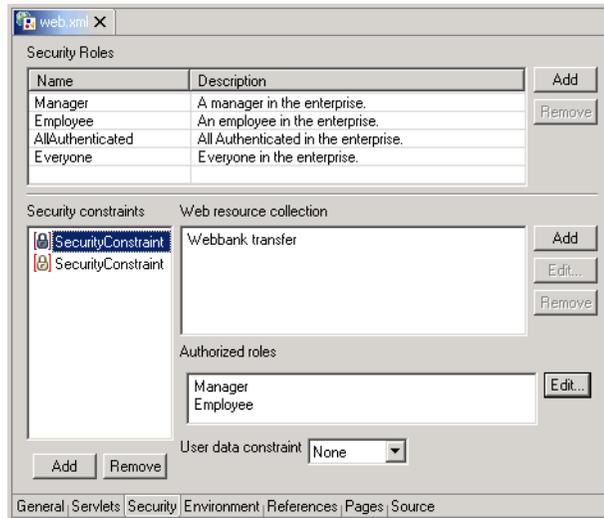


Figure 6-17 web.xml - Security window

12. Save the changes, then close the web.xml document.

6.5 Configuring Web module security using WebSphere Studio Application Developer

In this section, we will show how you can configure the authentication method and the security constraints using WebSphere Studio Application Developer.

1. Start WebSphere Studio Application Developer with the Webbank project.
2. First, we will define the authentication method. In the left-hand pane, select the **Navigator** view and expand the webbankWeb folder, then the webApplication and the WEB-INF folders.
3. Double-click the **web.xml** file. In the right-hand pane, you will see the Web application deployment descriptor configuration panel. Select the **Pages** tab.

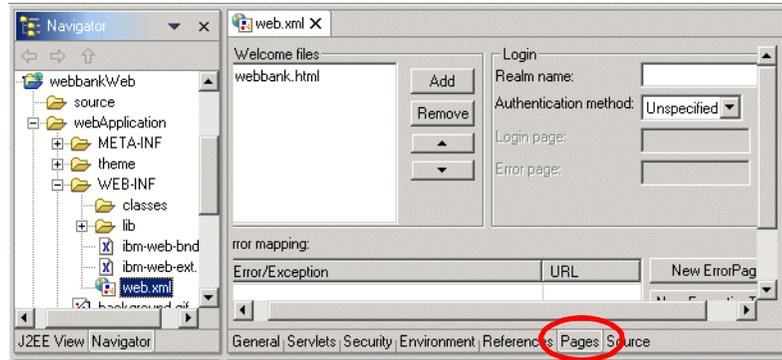


Figure 6-18 Authentication method pane

4. In the Login area, select the **Authentication method** drop-down menu and choose the method you want to use. The choices are Basic, Digest, Form and Client-Cert. For deployment into WebSphere, do *not* choose Digest.
5. Now enter a Realm name; in our example, we used Webbank Realm.
6. If you selected **Form** for the authentication method, you will have noticed that this enabled the Login page and the Error page input fields. You now need to define the pages you will use for this. The pages you choose must already be part of the Web module. Use the buttons to the right of each field to select:
 - Login page: /Login.html
 - Error page: /Login.html

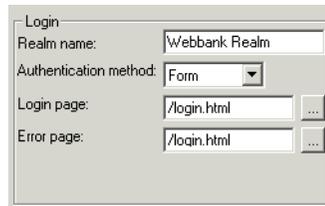


Figure 6-19 Completed form based login information

Now we will define a security constraint for the static file webbank.html and a second one for TransferServlet.

1. In the right-hand pane, switch to the Security tab.
2. Under the Security constraints area, click **Add**. You will see a new security constraint and a new Web resource collection created for you.



Figure 6-20 New security constraint

3. Select **New Web Resource Collection** and change the name to Webbank access.
4. Next to the resource collection area, click **Edit**. You will be presented with the **Web Resource Collections** entry window.
5. Next to URL Patterns, click **Add**. A new URL pattern will appear in the list. Select it and change it to / (forward slash on its own). Add a second URL pattern: /webbank.html.

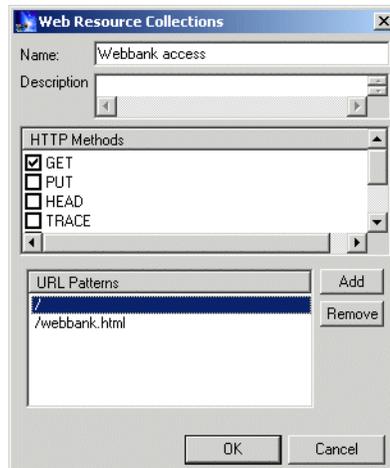


Figure 6-21 Completed Web Resource Collection window

6. Click **OK** to save the changes.
7. Now select the **Edit** button next to Authorized roles. You will be presented with the Select Auth Constraints entry window. Enter a description of Webbank access and select the role **AllAuthenticated**.



Figure 6-22 Webbank access role assignment

8. Click **OK** to save the changes.
9. For the TransferServlet, we will add a new security constraint and resource collection. Follow steps 8 to 13, using the following values from Table 6-1.

Table 6-1 Transfer Servlet security constraint information

Field name	Value
Web Resource Collection name	Webbank transfer
URL pattern	/TransferServlet
HTTP methods	GET, POST
Security Constraint description	Webbank Transfer constraint
Authorized roles	Employee, Manager

10. Save your changes to the Web module deployment descriptor, then close the file. The completed configuration should look like that shown in Figure 6-23 on page 131.

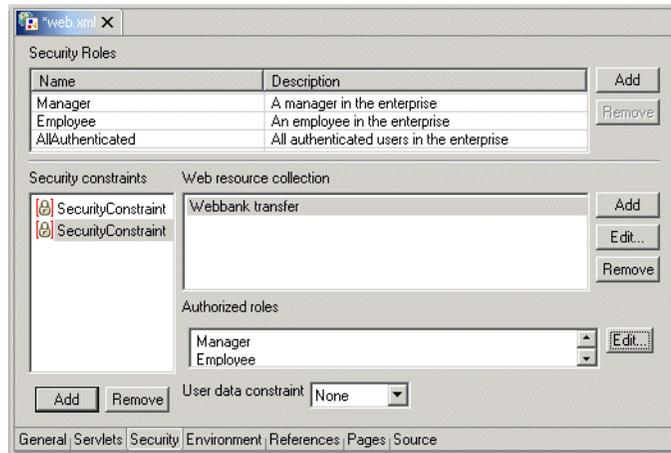


Figure 6-23 Completed Web security constraints

6.6 Form-based and Custom Login facilities

Customizable login facilities are available in WebSphere Application Server; form-based and custom login methods will be discussed here.

6.6.1 Form-based login

Form-based login, as described above, is defined by the Servlet 2.2 specification and is a requirement for J2EE 1.2 compliance. When a user requests a protected resource, they are first redirected to a login form. The user enters the details and submits the form; this triggers the WebSphere *FormLoginServlet*, which performs the authentication. If the authentication is successful, the user is presented with the originally requested page.

This form-based login is very simple. There are two entry fields and one Post action which must exist in the login form:

- ▶ *j_username* must be the field name for the user name.
- ▶ *j_password* must be the field name for the password.
- ▶ *j_security_check* must be the Post action for the Submit button.

The form used in the Webbank application is shown below.

Example 6-2 Login.html for Webbank application

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY background="background.gif">
<P align="center"><IMG src="CL-Logo.gif" width="358" height="53"
border="0"><BR>
You have been redirected to this page because the resource you try to access is
protected. Please supply a userid and password below. After successful login,
you will be automatically redirected to the page you originally requested.</P>
<FORM method="POST" action="j_security_check"><BR>
<TABLE border="0">
  <TBODY>
    <TR>
      <TD>Userid :</TD>
      <TD><INPUT size="10" type="text" name="j_username" maxlength="25"></TD>
    </TR>
    <TR>
      <TD>Password: </TD>
      <TD><INPUT size="10" type="password" name="j_password" maxlength="25"></TD>
    </TR>
  </TBODY>
</TABLE><BR>
<INPUT type="submit" name="action" value="Login"> <INPUT type="reset"
name="reset" value="Clear"><BR></FORM>
</BODY>
</HTML>
```

Logout

The Servlet 2.2 specification does not provide a logout facility. WebSphere extends the specification by providing a form-based logout facility. After logging out, the user is required to reauthenticate to gain access to protected pages. This logout form can be an HTML or a JSP page which has the Post action *ibm_security_logout*. This form must exist within the same Web application to which the user gets redirected after logging out. See the webbank.html file for the source code.

6.6.2 Custom login

In previous versions, WebSphere supported a Custom Login function which allowed you greater control over how the user is logged in. This allowed you to choose what information to receive from the customer and also what additional checks should be performed before authenticating the user. For example, it may

have been useful to verify that the user's subscription to your Web site had not expired before authenticating the user against your user registry. By providing a login helper class that could be extended, you were able to modify the default behavior of WebSphere during the authentication stage. In WebSphere V4, this facility is deprecated.



Securing EJBs

In this chapter, EJB security is discussed in detail. We will show how to use the Application Assembly Tool and the WebSphere Studio Application Developer to set security for the EJBs.

The following topics will be discussed:

- ▶ Method level permissions and role mapping
- ▶ Delegation policy settings and Run-As mode mapping

7.1 Securing EJBs

Enterprise Java Beans are essential parts of an enterprise application. EJBs implement the business and most of the application logic. They have access to sensitive data, and it is very important to understand the security applied to these enterprise resources. Apart from the Web resources, EJBs should also be carefully secured.

If global security is enabled and the EJBs have no methods at all configured with security, then the default is to grant access to the EJBs' methods. If global security is enabled and at least one method has a security constraint, then the request to the EJBs is denied. This kind of behavior is different from that of the Web modules' components. The following table summarizes this:

Table 7-1 Securing EJBs

Security enabled No method permission	EJB granted
Security enabled One method is protected, the other has a default configuration	EJB denied

For more information on how to secure EJBs' method permissions specified in the EJB specification 1.1, refer to the following URL:

<http://java.sun.com/products/ejb/docs.html>.

Figure 7-1 depicts the User-Group-Role-EJB Method mapping. Roles can include users and/or groups; groups can include users, and roles are then mapped to EJB Methods.

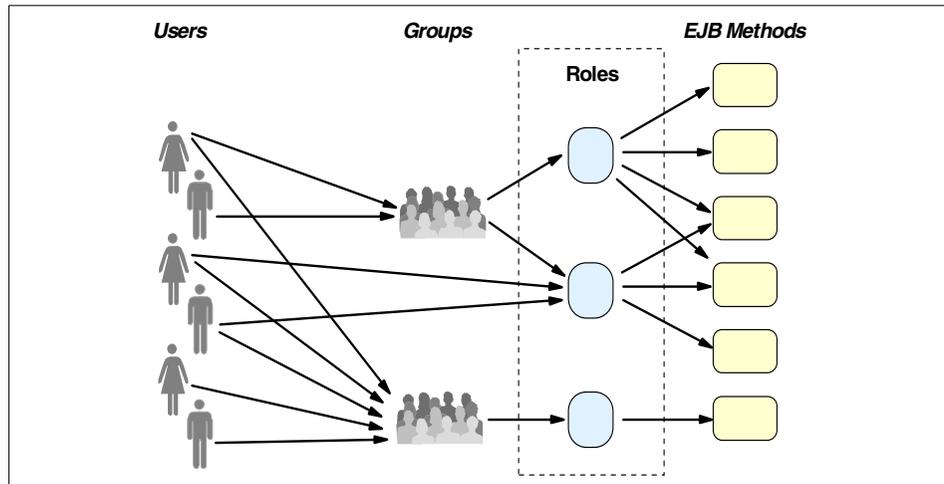


Figure 7-1 Method mappings

The following table summarizes the EJB security settings, the place where the settings are stored and the tools to use to change the settings.

Table 7-2 Creating the method permissions for EJBs

Security Information	Storage	Tools
For each EJB, assign each of its methods to one or more roles	EJB DD <code>ejb-jar.xml</code>	AAT
Optionally, set up Security Role References (Run-As Mapping)	EJB DD <code>ejb-jar.xml</code>	AAT, AC
Configure the delegation policy, Security Identity (Run-As Mapping)	EJB DD <code>ibm-ejb-jar-ext.xmi</code> , Repository DB	AAT, AC

7.2 Assigning methods to roles

The EJB method permission maps one or more security roles to one or more methods that a member of the role can invoke.

You may want to use the table shown in Figure 7-2 for assistance.

The first column defines the methods implemented in the EJB, and the headers above the other columns list the roles which can access the EJB methods. The cells in the table specify whether or not the role has access to a certain method. In order to determine the authorization for a method, find the row for the method, then find the column for the role name. If the cell in the intersection is empty, then the role cannot run that method, but if the cell is checked () then the role can run the method.

Method	Method Type	Manager	Employee	AllAuthenticated	Everyone	Consultant
<input type="checkbox"/> BranchAccount						
*	Unspecified	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
findByNameLike(java.lang....	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
findByPrimaryKey(itso.web...	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getEJBMetaData	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getHomeHandle	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBalance	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBalanceCent	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBranchAddress	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBranchID	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBranchName	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getEJBHome	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getHandle	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getPrimaryKey	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
isIdentical(javax.ejb.EJB...	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> CustomerAccount						
*	Unspecified	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
findByPrimaryKey(itso.web...	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getEJBMetaData	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getHomeHandle	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getAccountNumber	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBalanceCent	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getCustomerID	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getCustomerName	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getEJBHome	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getHandle	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getPrimaryKey	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
isIdentical(javax.ejb.EJB...	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> Transfer						
*	Unspecified	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Consultation						
*	Unspecified	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 7-2 Method permissions for Webbank

The security roles must be defined, and the methods must be in the enterprise bean's remote or home interface.

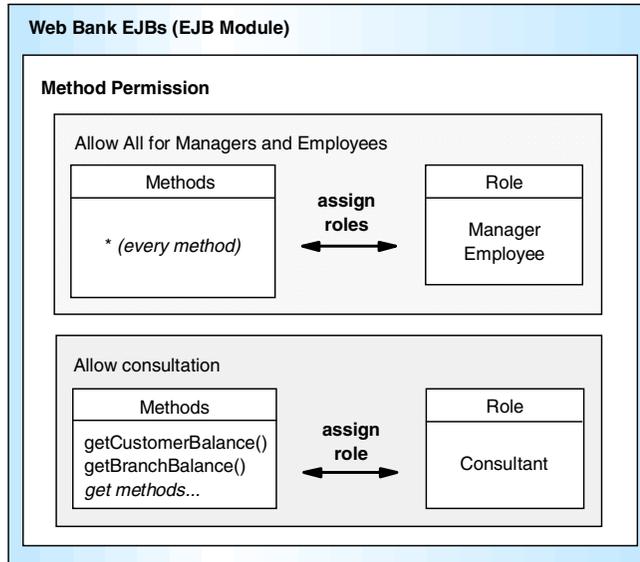


Figure 7-3 Method Permission

Two Method Permissions are defined in Figure 7-3: *Allow Teller to Perform Transaction* and *Allow Manager to Handle Accounts*.

To assign methods to roles, the roles must exist. The following user roles should be defined:

- ▶ Manager
- ▶ Employee
- ▶ Consultant

beside the special roles:

- ▶ All Authenticated
- ▶ Everyone
- ▶ DenyAllRoles (this will be automatically defined during installation, unless you want to predefine it)

7.2.1 Configuring method permissions using AAT

The following screen capture shows the roles in the Application Assembly Tool (AAT), required for the Webbank sample application.

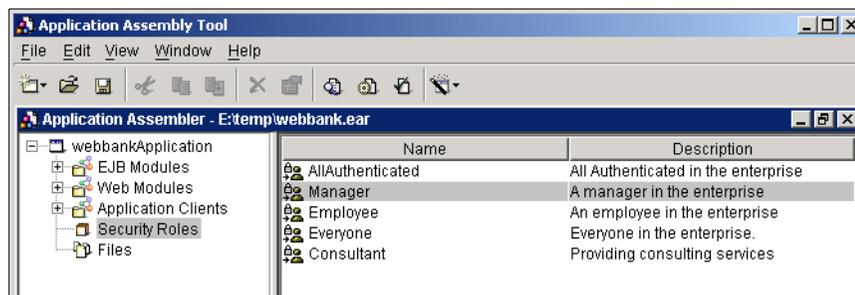


Figure 7-4 Security roles

Security roles must be defined for the EJBs; the roles can be defined in the EJB module or on the application level.

The following steps will show how to set up a role in the EJB module using the Manager role as an example.

1. Launch WebSphere's Application Assembly Tool (AAT) and open the Webbank application.
2. Expand the **EJB Modules** -> **webbankEJBs** folder under webbankApplication.
3. Right-click **Method Permissions** and select **New**. The following window appears (see Figure 7-5 on page 141).

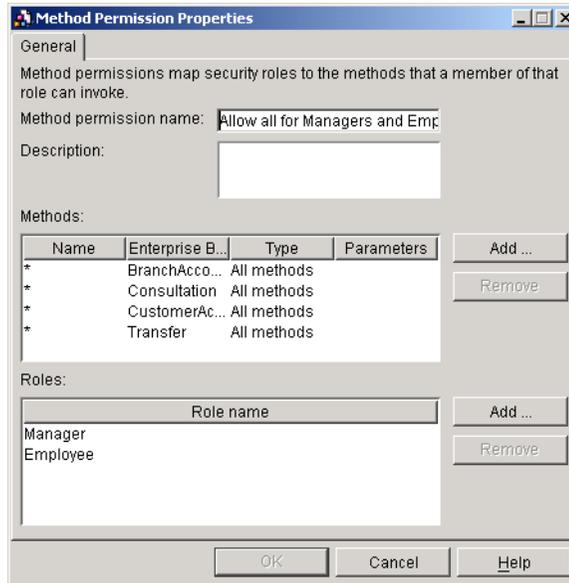


Figure 7-5 New method permission

A permission needs a name and description for the methods which are allowed to be executed.

4. Provide the details for this entry:

Method permission name: Allow all for Managers and Employee

Description: Managers and Employee can perform any transaction without restriction

5. Click **Add** beside the methods list; a new window appears listing all the EJBs and their methods. Select the methods that the role is allowed to execute. Notice that the * (asterisk) represents all methods which belong to that group.

Note: You can select multiple methods by using the Control key on the keyboard together with the left mouse button.

You can select a range of methods by using the Shift key on the keyboard combined with the left mouse button. Select the first entry, then hold down the Shift key and select the last entry.

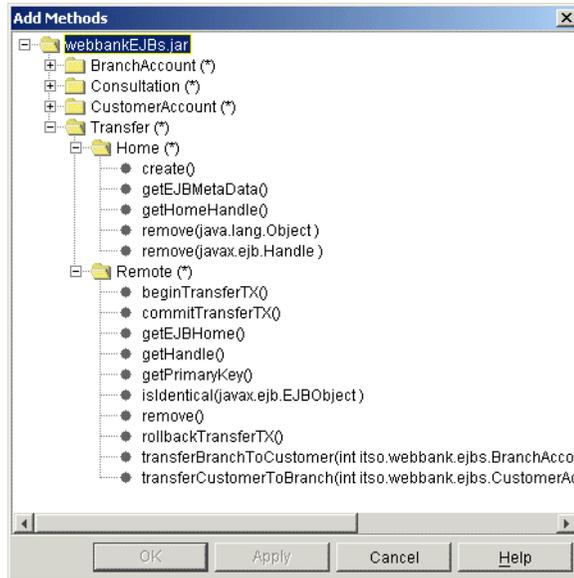


Figure 7-6 Selecting EJB methods

Methods for this role are:

- BranchAccount(*)
 - Consultation(*)
 - CustomerAccount(*)
 - Transfer(*)
6. Finally, add the Role by clicking **Add** on the left side of the roles list. A new window appears with all the roles; select the **Manager** and **Employee** roles. Click **OK**.

Note: You can implement multiple selections when selecting the items for the method permissions, by combining the left mouse button with the Ctrl key.



Figure 7-7 Selecting the role

These are the steps necessary in order to assign authentication to methods. Another method permission is required for consultation:

Method permission name: Allow read for consultants

Description: Allows the read methods for the consultants.

Roles assigned: Consultant

For a detailed list of roles, see Figure 7-2 on page 138. The rightmost column of the table shows the permitted methods for the *Consultant* role.

7.2.2 Configuring method permission with WebSphere Studio Application Developer

WebSphere Studio Application Developer (WSAD) supports setting security during development time. The application developer can set all the method permissions for the EJBs in WebSphere Studio Application Developer, though the process is a bit different than with the Application Assembly Tool (AAT).

The following steps will show you how to set the method permissions for the EJBs.

1. Launch WebSphere Studio Application Developer with the Webbank project.
2. Switch to the J2EE view, or open it from the menu (**Perspective open -> J2EE**).
3. Select **EJB Modules -> webbankEJBs** then switch to the **Security** tab. In the upper table, the security roles are listed, while in the lower table are the method permissions defined for the EJB methods.

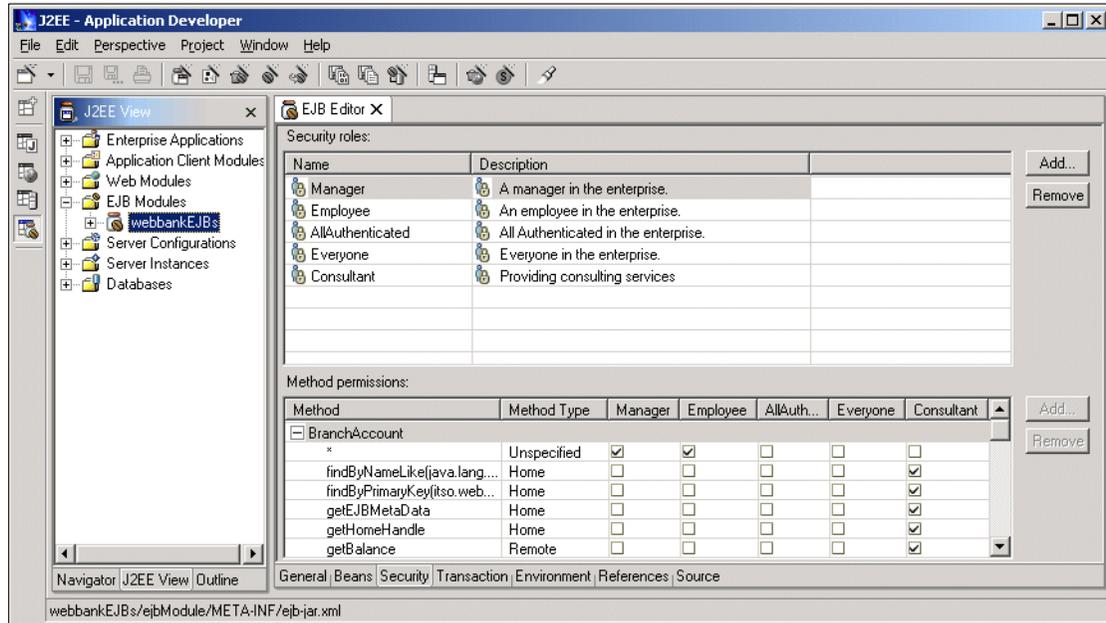


Figure 7-8 WebSphere Studio Application Developer method permission

4. Select the EJB in the lower table for which you want to add a method permission, then click **Add** next to the table.
5. A new window appears: *Add Method Permission*. Select the method type, select the method, then select the desired role and click **OK**. A new entry will appear below the EJB with the chosen method name and method type, and the permissions will be shown as checkboxes in the role columns.
6. In order to change method permissions for a given method, you must select the line with the method name, then give or remove permission by selecting or deselecting the checkbox below the roles, as in Figure 7-9.

Method	Method Type	Manager	Employee	AllAuth...	Everyone	Consultant
BranchAccount	Unspecified	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
*						
findByNameLike(java.lang....	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
findByPrimaryKey(itso.web...	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getEJBMetaData	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getHomeHandle	Home	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getBalance	Remote	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 7-9 Method permission in detail

7. Assign the listed methods from Figure 7-2 on page 138 to the roles; you should see the same result as in the table.
8. Save the changes using **File -> Save EJB Editor** or **Save all**, then close the file.

7.3 Setting up security role references

Security role references are used to shield the developer from all deployment details. A security role reference links the name of a role used in a module to the corresponding name of a role in the encompassing application. The application assembler performs the mapping of the role reference defined in the source code to the role name defined in the enterprise application role by using the role link tag in the deployment descriptor.

Example 7-1 Sample for isCallerInRole method

```
public Object doTransfer(EJBContext con) {
    //obtain the user name
    if (con.isCallerInRole("Employee") && transferAmount>allowedAmount) {
        // do the transfer
    } else
        // send alert message that the amount is not allowed
    }
}
```

The programmer has defined the role *Employee* during development. The deployer maps *Employee* to the organizational structure of the bank where *Employee* was once defined.

The following step-by-step guide is only for demonstration purposes and has no effect on the application.

1. Launch AAT, then open the webbank application.
2. Expand the *TransferSessionBean* folder (**webbankApplication -> EJB Modules -> Session Beans -> TransferSessionBean** folder).
3. Right-click **Security Role Reference**, then select **New** and type in the following details in the *New Security Role Reference* window (see Table 7-3 on page 146).

Table 7-3 Security Role Reference

Field name	Field value
Name	Employee
Link	Employee
Description	This role cannot do a transfer higher than a specified value

The two roles are linked through the name and the provided link; for clarity, a description can be provided. The panel should look like that shown in Figure 7-10.

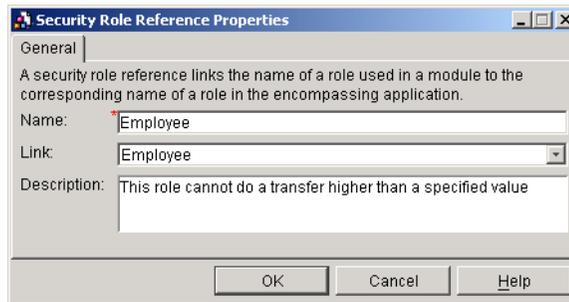


Figure 7-10 Security role reference in AAT

4. Confirm with **Apply** or **OK**.

The Employee (Source Code) is mapped to the Employee (Enterprise Application) and later on the Employee is mapped to the role defined in the user registry, as shown in Figure 7-11.

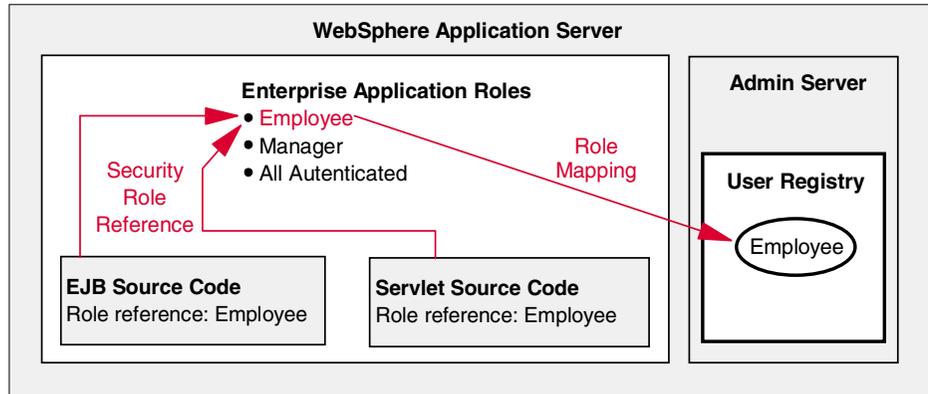


Figure 7-11 Security role reference overview

The security role reference is configured in AAT, whereas the role mapping is performed using the Administrator's Console.

7.4 Configuring the delegation policy

The delegation policy determines the identity under which the method should be executed. If a downstream method call is necessary, it occurs in this identity. The delegation security information is stored in the deployment descriptor and can be changed with AAT. Because this information is stored in the deployment descriptor, there is no need to access the repository, because the information is loaded into the memory at runtime.

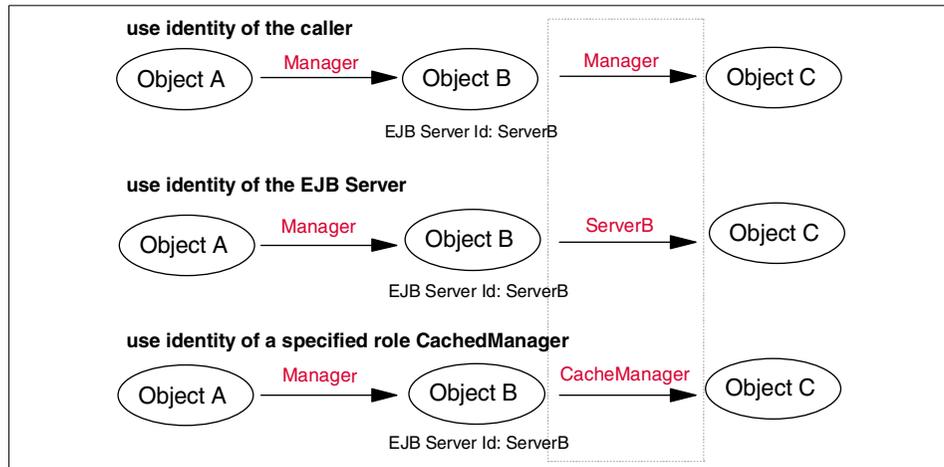


Figure 7-12 Delegation

The delegation specifies the security identity under which the EJB method will be executed. This identity can be that of either the caller, the EJB server, or it can be a specified identity.

The following sample shows the three methods with three different delegation modes. In the case of “use identity specified,” a specific role must be used.

Table 7-4 Delegation flow

Method	Delegation Mode	Specified Role
getAccountBalance	use identity of the caller	-
doTransaction	use identity of the system	-
getBankNews	use identity specified	CacheManager

Looking at Figure 7-12, let us assume that the user, who is in the *Manager* role, accesses an EJB (Object B) from a servlet (Object A), which is a kind of interface to another EJB (Object C).

- ▶ If the *manager* calls the method *getAccountBalance*, the delegation mode is set to **caller's identity**; Object C is then called with the identity of the manager.
- ▶ If the manager calls the method *doTransaction* as *manager* and the delegation mode is set to **system**, then Object C is called with the EJBServer identity, which is *Server B*.
- ▶ If *manager* calls *getBankNews*, the **specified identity** (*CacheManage*) is used to run the method.

These are requirements you may need for applications that need a consistent return value of *getCallerPrincipal()* across a chain of calls between EJBs. You may also want to use these requirements for performance aspects by calling using always the same ID.

Note: If you set Run-As mode to **use identity specified** then the identity must be defined through the Run-As mapping.

7.4.1 Setting up delegation policy (Run-As mode) using AAT

The following step-by-step guide is only for demonstration purposes and has no effect on the application.

1. Launch AAT, and open the webbank application.
2. Expand **webbankApplication -> EJB Modules -> webbankEJBs -> Session Beans -> ConsultationSessionBean**, then select **Method Extensions**.
3. Select the desired method on the right side of the list at the top.
4. Select the **Advanced** tab, enable Security Identity, then choose from one of the following options for the Run-As mode:
 - Use Identity of caller (this is the default setting)
 - Use Identity of EJB Server
 - Use Identity assigned to a Specific Role and choose one of the roles in the pull-down menu

For our example, configure the following delegation policy as shown in Table 7-5.

Table 7-5 Security role reference

Name	Value
Bean	ConsultationSessionBean
Method	getCustomerBalance
Security Identity	enabled
Run As	Use Identity assigned to a Specific Role
Role	Manager
Description	This call is always made in the role Manager for performance reasons

Figure 7-13 shows a screen capture from AAT including the delegation policy set for the *getCustomerBalance* method of the *ConsultationSessionBean*.

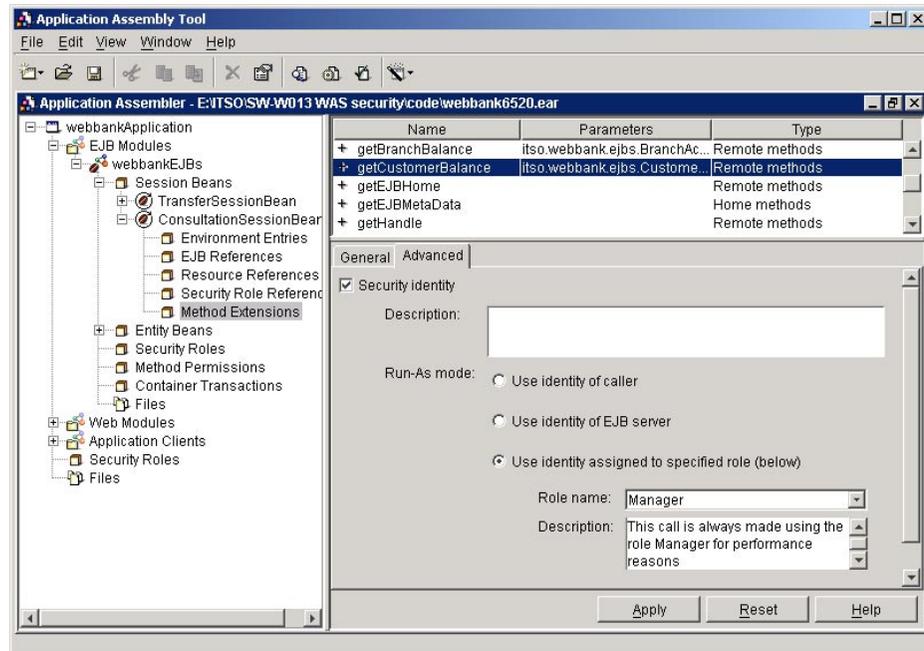


Figure 7-13 Delegation policy set in AAT

The application will require that you set the Run-as mapping either during the deployment or using the Security Center from the Administrator's Console. You will find the steps used to do this in "Run-As mapping using the Administrator's Console" on page 151.

7.4.2 Setting up delegation policy (Run-As mode) using WebSphere Studio Application Developer

J2EE application developers can also define Run-As mode mappings using the WebSphere Studio Application Developer. The steps you must follow to obtain the same results as in the previous section, using AAT, are described below.

1. Launch WebSphere Studio Application Developer with the webbank project.
2. Switch to the J2EE perspective, then select the **J2EE View** tab.
3. Expand the **EJB modules** folder, right-click the **webbankEJBs** item, then select **Open with -> EJB Extension Editor**.

4. The *EJB Extension Editor* document appears in the right-hand pane; switch to the **Methods** view (you do not have to switch if you have just opened the window, because it is the default tab upon opening).
5. Expand the Remote Methods folder (**webbankEJBs -> Consultation -> Remote Methods** folder), then select the **getCustomerBalance(...)** method. On the right-hand side panel, you will find all the settings for this method, as it was described in the previous section under the AAT.
6. Set the *Run as* mode to **Use identity assigned to specific role (below)**.
7. Then set the Role name to **Manager** using the drop-down menu.
8. Save the document, then close it.

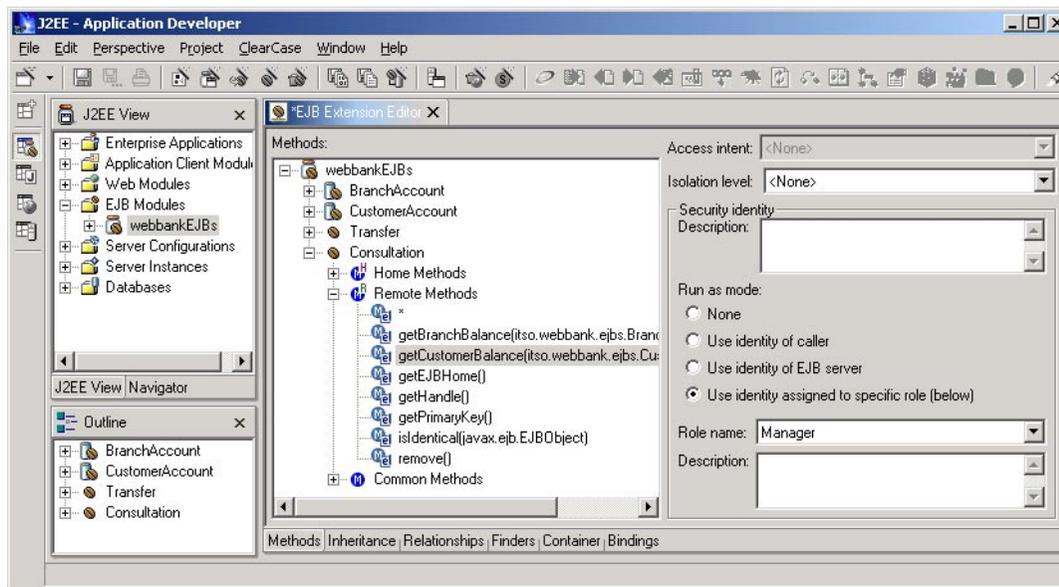


Figure 7-14 Setting up the Run-As mode using WebSphere Studio Application Developer

The application will require you to set the Run-as mapping either during deployment or by using the Security Center from the Administrator's Console. You will find these steps described in "Run-As mapping using the Administrator's Console" on page 151.

7.4.3 Run-As mapping using the Administrator's Console

The Run-As mode was assigned in the AAT tool; the mapping must be performed using the Administrator's Console. At this point, the application role is mapped to a specific user in the user registry.

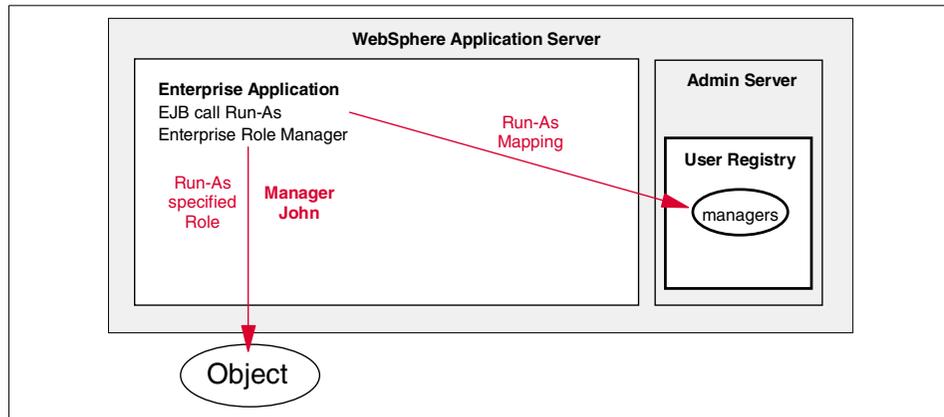


Figure 7-15 Delegation

The delegation calls an object with a specific enterprise role manager. To do so, the role must be mapped to a role from the user registry (manager).

1. Open the Security Center by clicking **Console -> Security Center**. In the Security Center, switch to the **Run As Role Mapping** tab.
2. Select your Enterprise Application (webbankApplication) and click **Edit Mappings....** A new window appears .

For a specific enterprise application, the Run-As role mapping must be customized.

3. Select the role on the left, then click **Select**.
4. A window appears, prompting for a user name and password for mapping. The user must be defined in the selected role, or must be a member of a group which is defined in the selected role. Enter the user name and the password, then click **OK**. In our example in Figure 7-15, *Manager John* must be in the *Manager* role.
5. Click **OK** to finish the mapping.

Once the specified identity (role) Run-As mode has assigned one of the methods in an EJB, the Run-As mapping must be defined for the specified role.

7.4.4 Run-As mapping during deployment

The configurations previously discussed were performed during application assembly and development time, but these configurations can also be set at deployment time. The following description will illustrate the step from the enterprise application installation when Run-As mapping occurs.

Once you have selected the enterprise application archive file, the second window, *Mapping EJB RunAs Roles to Users*, will ask for the Run-As mappings, as shown in Figure 7-16.

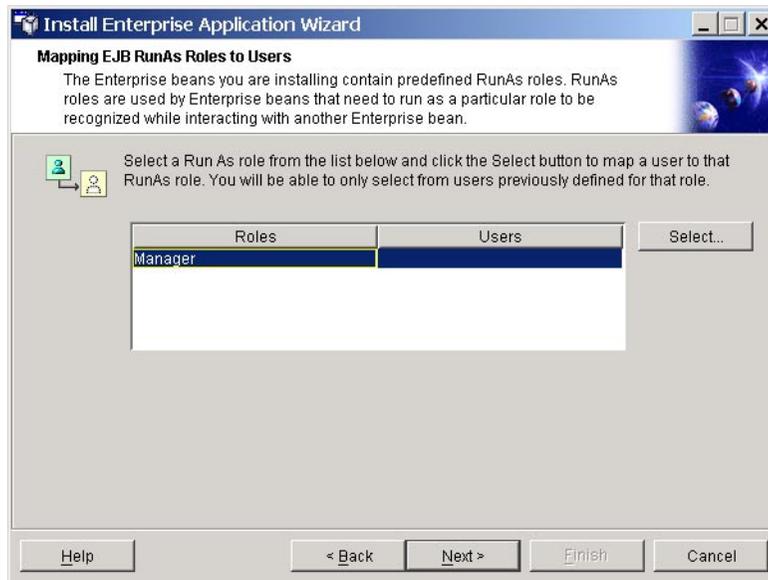


Figure 7-16 Run As role mapping during EAR installation

At this point, everything works the same way as when using the Administrator's Console for Run-As mapping (see "Run-As mapping using the Administrator's Console" on page 151).

7.5 Topology considerations

Enterprise beans and their clones (a number of application server instances) have separate identities. To avoid security breaches, every bean must be protected by configuring resource security for the bean and including it in a secured enterprise application.



Securing J2EE clients

This chapter discusses security for the different types of Java clients. By *Java clients* we mean clients that implement EJB clients, which then connect to the application server.

First, we will explore the Secure Association Service, which is a fundamental element of WebSphere Application Server security.

Next, we will describe the programmatic login, which is tightly related to Java clients.

Finally, we will discuss the different types of clients which may be involved.

8.1 J2EE clients

J2EE clients is a general term which describes a broad range of clients (not just specific ones) connecting to J2EE applications.

The simplest way to describe the clients would be to say that they are programs which implement EJB clients in order to connect to EJBs.

There are three different application clients which can be identified:

- ▶ J2EE application client
- ▶ Applet client
- ▶ Java thin application client

Others, such as servlets, JSPs, or even EJBs, can also implement EJB clients.

For more information about Java clients, please refer to the WebSphere Infocenter's section 4.7 (Java clients) at:

<http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>.

8.2 The Secure Association Service (SAS)

The requests from clients to Enterprise JavaBeans are sent as RMI/IIOP messages, using the Object Request Broker (ORB), to the server that hosts the Enterprise beans. As part of every such request and response, the ORB invokes the Secure Association Service (SAS) on both the client and the server sides.

Figure 8-1 shows the interaction between an EJB client and the EJB when global security is enabled.

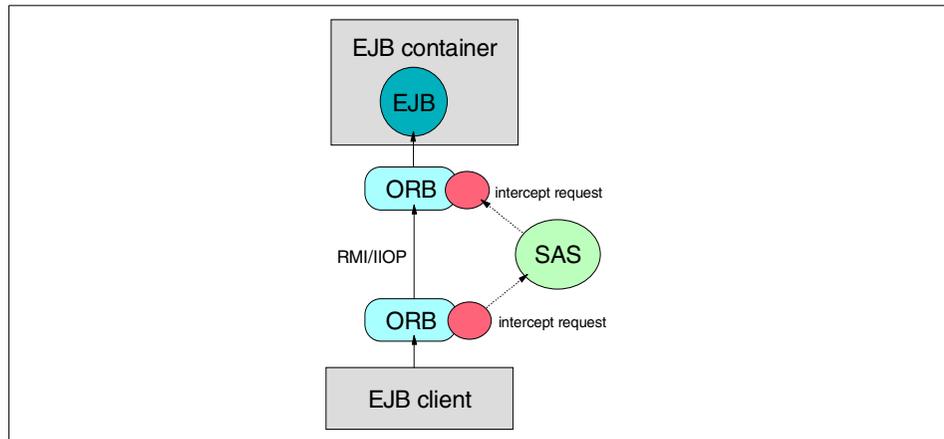


Figure 8-1 SAS interception

As part of every request and response, the ORB invokes the SAS on both the server and client sides. SAS intercepts the request coming from the client, collects the security credentials and attaches those to the request, then sends the request. On the server side, SAS intercepts the request again, extracts the security context, authenticates the client, then sends the request to the EJB container. The EJB container authorizes the request, then serves the EJB request. The response goes back along the same route through the SAS interceptors. The communication between the two ORBs is secured by SSL.

How does the application know about security?

The clients do not have to handle security; they can implement a programmatic login, but security must function without any written code. In order to use security, the client must know the security settings for the ORB. These settings are stored in the SAS properties file (sas.client.properties). You can specify the file for the JVM using the following directive:

```
com.ibm.CORBA.ConfigURL=URL of the properties file
```

which should look like this in your case:

```
java -Dcom.ibm.CORBA.ConfigURL=file://properties/sas.client.props
itso.webbank.client.WebbankThinClient
```

The properties for the ORB will be picked up from the specified file; the client can then make the secure connection.

When should SAS programming be used?

SAS programming is useful when applications must log in programmatically or if they have to manipulate the credentials at runtime. Manipulating the credentials is necessary when specific methods require credentials different from the ones originally owned by thread. The SAS programming interfaces are based on the CORBA Security Service specification; for more information, refer to the OMG Web site at <http://www.omg.org> and obtain the CORBA Security Service specification. Programmatic login is covered in “Programmatic login” on page 161.

8.2.1 SAS on the client side

When an EJB client invokes a remote method, SAS interceptors are called to perform the following process:

1. Establish an SSL connection between the two ORBs.
2. Establish a secure association between the client and the server.
3. Send the request to the server.
4. Receive the response from the server.

Two information sources are used to establish the SSL connection between ORBs:

- ▶ The Interoperable Object Reference (IOR)
- ▶ The SAS properties file on the client side

The server’s IOR provides the following information:

- ▶ Server TCP/IP address
- ▶ Server TCP/IP port
- ▶ Server SSL port
- ▶ Server security name; for a local operating system, this follows the format *DOMAIN/server_id* . For LTPA, the format is *LDAP_HOST_AND_PORT/server_id*.
- ▶ Quality of protection (QOP) required: high, medium or low.

The properties file

The configuration file for SAS on the client side is the `sas.client.props` file.

Note: Only a few settings are documented here; for more information about SAS properties, refer to the WebSphere Infocenter, section 5.7.5 (SAS properties reference) at: <http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>, or read the comments in the SAS properties files provided within WebSphere Application Server AE V4.

A short list of client's SAS settings follows:

- ▶ `com.ibm.CORBA.securityDebug` - used to enable/disable debugging for SAS; valid values are *true* or *false*.
- ▶ `com.ibm.CORBA.securityTraceOutput` - the file which holds the debug trace for SAS, for example: `logs/sas_client.log`.
- ▶ `com.ibm.CORBA.standardPerformQOPModels` - this is *Quality of Protection*; the valid values are *high* (default), *medium*, and *low*. These values entail different key bit lengths (high=128-bit, medium=40-bit, low=no encryption).
- ▶ `com.ibm.CORBA.loginSource` - the source of the user ID and password strings; the valid values are *prompt* (default), *properties*, *keyfile*, *stdin*, and *none*.
- ▶ `com.ibm.CORBA.loginTimeout` - the timeout, after which SAS removes the login prompt; the value is set in seconds and the default is 300 seconds. After timeout, SAS removes the login prompt and the request is handled with no security.
- ▶ `com.ibm.CORBA.authenticationTarget` - this value determines the authentication method used to establish credentials. The valid values are *basicauth*, *localos*, and *ltpa*. For a pure Java client, only the *basicauth* is supported.
- ▶ `com.ibm.CORBA.securityEnabled` - this indicates the security enabled status; it should be set to *true*.
- ▶ `com.ibm.ssl.keyStore` - this defines the key store file for SAS, for example: `keys/DummyClientKeyFile.jks`. By default, it uses the dummy key store provided with WebSphere. In a production environment, it is strongly recommended to replace the dummy key store and trust store files with new store files.
- ▶ `com.ibm.ssl.trustStore` - this defines the trust store file for SAS, for example: `keys/DummyClientTrustFile.jks`.

Note: for information about how to replace the dummy key store and trust store files, refer to the Section 11.1.1, “The Demo Keyring” on page 217.

The login information can be collected using several methods; the method is specified in the SAS client properties file with the *com.ibm.CORBA.loginSource* setting. The following authentication mechanisms are available:

- ▶ *prompt* - this provides a graphical window on the screen to collect the user ID and password. Pure Java clients must call **System.exit(0)** at the end of the program to properly finish the process.
- ▶ *standard input (stdin)* - only supported for pure Java clients, the program prompts for user ID and password using a non-graphical console prompt.
- ▶ *properties* - the user ID and password are retrieved from the following two properties: *com.ibm.CORBA.loginUserid*, and *com.ibm.CORBA.loginPassword*. With this method, the client also has to specify the *realm*. You can specify the realm using one of the following settings:

- a. `com.ibm.CORBA.loginUserid=userid`
`com.ibm.CORBA.principalName=REALM/userid`
- b. `com.ibm.CORBA.loginUserid=REALM/userid`

Where REALM and userid are set for the client.

- ▶ *key file* - the *user ID* specified in the *com.ibm.CORBA.loginUserid* property and the *realm* name from the IOR are used to extract the user ID and password from the key file specified in the *com.ibm.CORBA.keyFileName* property.
- ▶ *none* - this is only available if programmatic login is used in the client code.

8.2.2 SAS on the server-side

When an EJB client invokes a remote method on the server, the following happens on the server side:

1. SAS intercepts the request and performs the necessary security checks.
2. The user is authenticated.
3. The server invokes the method.
4. The response is sent back to the client.

On the server side, it is also necessary to configure the SAS runtime. The settings from the administration server are propagated to the repository, then eventually to the SAS server properties file (*sas.server.props*).

The properties file

The configuration file for SAS on the client side is the `sas.server.props` file; it can be found in `<WebSphere install directory>\properties`. The SAS properties for the server are very similar to the client properties, except for some server-specific settings. The SAS settings for the server are very well documented in the WebSphere Infocenter, found at <http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>. More information can be found in the SAS properties file, as comments provided within WebSphere Application Server AE V4.

For more information about SAS tracing, refer to “The Secure Association Service (SAS)” on page 449.

8.3 Programmatic login

Please note that though programmatic security and programmatic login are closely related, they are two different topics. The other closely related topic is that of Single Sign-On, which is covered in Chapter 14, “Single Sign-On” on page 393.

The EJB client application must collect information about the clients to authenticate them. User authentication can be performed in two separate ways:

- ▶ Client-side login
- ▶ Server-side login

Note: No authentication occurs unless WebSphere global security is enabled.

Clients are also required to log in when accessing Web resources in the Web module. It is also possible to use a customized login form, page or servlet for user authentication. Customized login for the Web module is covered in Chapter 10, “Programmatic security” on page 201.

8.3.1 Client-side login

Client-side login is useful when the client needs to log a user into the security domain, but does not need to use the authentication data itself. Client-side login collects the login information and sends it to another program for actual authentication. Practically thin Java application clients, J2EE application clients, Java applets, where the client is running on the client side, all use client-side login.

The client-side login process flow is as follows:

1. The user invokes a request with the client.
2. The client collects the login information (user ID, password) from the user.
3. The client places the user's authentication data into the ORB-related data structure called the *security context*.
4. The client invokes a method on the server by sending a request.
5. The server processes the request and extracts the authentication data from the security context.
6. The next step is based on the result of the authentication:
 - a. If the authentication was successful, the server grants the request and returns the security credentials for further use.
 - b. If the authentication was unsuccessful, the server denies service.

The key in this process is that the client *sets the security context* for the ORB.

The following code excerpt includes some of the steps described above.

Example 8-1 Client-side login from WebbankClient.java

```
...
String userid = args[2];                                     (2)
String password = args[3];
...
LoginHelper loginHelper = new LoginHelper();
try {
    org.omg.SecurityLevel2.Credentials credentials = loginHelper.login(userid,
password);                                                  (3)
    loginHelper.setInvocationCredentials(credentials);       (6a)
...
} catch (...)
...

```

Note: Steps 1, 4 and 5 are not shown in the code example because they are hidden from the developers.

Step 1 is a user interaction.

Step 4 is performed by the *_LoginHelper* provided by the ORB security support class.

Step 5 is performed on the server.

For more information about the LoginHelper class, refer to the WebSphere Infocenter, found at <http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>.

There is a client-side programmatic login sample provided with the redbook; for more information about this sample, refer to “Client-side login” on page 161.

8.3.2 Server-side login

Server-side login is used when the application has to log the users into the security domain and must use the authentication data itself. The server-side login collects the authentication data and performs the authentication. Server-side programs, servlets, JSPs, where the program is running on the server, all use the server-side login.

The server-side login process flow runs as follows:

1. The user request triggers a servlet.
2. The servlet collects the user authentication data (user ID, password); this can happen in many different ways.
3. The servlet presents the request to the server.
4. The server processes the request, then extracts the authentication data from the context and performs authentication.
5. The next step is based on the result of the authentication:
 - a. If the authentication was successful, the server grants the request.
 - b. If the authentication was unsuccessful, the server denies service.

The key in this process is that the code on the server side (servlet) performs the authentication. Furthermore, the authentication is performed for each request; no credentials are stored for further use.

The server program is responsible for extracting the authentication data, inserting it into the CORBA data structure and authenticating the user. WebSphere provides a utility class, ServerSideAuthenticator, which helps to perform the programmatic login. For more information about the ServerSideAuthenticator class, refer to the WebSphere Infocenter, found at <http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>.

The following code excerpt includes some of the steps described above.

Example 8-2 Server-side login from WebbankServletClient.java

```
...
String userid = request.getParameter("userid");           (2)
String password = request.getParameter("password");
...
boolean forceAuthentication = true;
ServerSideAuthenticator serverAuth = new ServerSideAuthenticator();
try {
    org.omg.SecurityLevel2.Credentials credentials = serverAuth.login(userid,
password, forceAuthentication);                          (4)
...
} catch (...)
...

```

Note: Steps 1, 3, 5 are not shown in the sample code, because they are hidden from the developers.

Step 1 is a user interaction.

Step 3 is performed by the core servlet code.

Step 5 is performed by the server.

Running the WebbankServletClient sample

The *WebbankServletClient* is a very simple EJB client embedded in a servlet. This example will use server side authentication. The following steps will show how to use the sample:

1. Access the Webbank application with a Web browser, using `http://<server name>/webbank`.
2. The application will ask for the user ID and password. Type in a user ID and a password for an employee (the user must be in the Employee group).
3. Access the following page with the Web browser: `http://<server name>/webbank/webbankservletclient.html`.
4. Fill out the form fields:
 - The user name should be that of a user in the Manager group.
 - The password is the password for the user.
 - The server's address is the IP address for the WebSphere Application Server. You can also specify the port number.
 - The new balance is a number which will be set for the branch account.

Make sure that you are using a different user than the one you have used to login the webbank application.

The purpose of this sample is to show that the server will use a different user ID from the original one to access the EJB. Normally, a user from the Employee group cannot change the branch account balance, but with this sample the servlet picks up a new user ID and accesses the EJB with it, which has the proper privileges.

8.4 J2EE application client

J2EE application clients are simple Java applications running in a J2EE environment. The J2EE application launcher provides all the necessary elements for the runtime, including the security features.

The Java application does not require any special programming from the security point of view; this is provided by the J2EE runtime environment.

8.4.1 Webbank J2EE client

The Webbank sample application uses a simple Java application with a graphical interface which can get the balance for the accounts. If you check the code, you will find that the application does not use any special code or API to access EJBs or to perform security.

Following are the additional steps the user has to follow in order to establish security:

1. Use the following command to launch the J2EE client:
`1 launchclient webbank6520.ear -CCBootstrapHost=<WebSphere AE server>`
2. When security is set for the enterprise application, the first window on the client will ask for user authentication (see Figure 8-2).



Figure 8-2 J2EE client challenge

3. Type in the user ID and password; this only requires a user from the Employee group, because it will only invoke a Read method.
4. Click **OK**.
5. From this point on, security is established and everything works as without security.

The client launcher sets the security properties for the application and specifies the SAS properties. The default property for login is a prompt for a user ID and password, so the SAS interceptor pulls up a window for these fields. The launcher behaves as a shell and provides all the necessary functions required for a J2EE client, in our case, the security features.

Note: The unauthorized access to the server's objects throws an exception. Exception handling for the unauthorized access has not been implemented, so you will see the classic Java stack trace on your console where you are trying to log on with the wrong user name and password.

8.5 Java thin application clients

A very simple EJB client application is provided with the book, focusing only on introducing the Java thin application client's security features.

The application is called *WebbankThinClient*. It uses the *LoginHelper* class provided with WebSphere V4.

LoginHelper is a supplemental class, implementing several methods to help programmatic login; these methods are:

- ▶ *org.omg.SecurityLevel2.Credentials login(String userid, String password)*

This sets the authentication data on the thread of execution.

- ▶ *void setInvocationCredentials(org.omg.SecurityLevel2.Credentials invokedCreds)*

This method sets the specified credentials as the invocation credentials. Any method invoked after this will be executed under the authority of the credentials.

- ▶ *org.omg.SecurityLevel2.Credentials getInvocationCredentials()*
This method retrieves the credentials under which the current method is being executed.
- ▶ *String getUserName(org.omg.SecurityLevel2.Credentials creds)*
This returns a human-readable user name attribute of the specified credentials.

For more information about the LoginHelper, refer to the WebSphere Infocenter, found at

<http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>.

Other resources are also required to support security for thin application clients:

- ▶ The file `sas.client.props` defines the security settings for SAS on the client side. The file can be found under the properties directory. It is a copy of the original file from the `<WebSphere instal directory>/properties` directory.
- ▶ Key store files and trust store files are also required for the SSL connection over IIOP. The two files are copies from the `<WebSphere instal directory>/etc` directory: `DummyClientKeyFile.jks`, `DummyClientTrustFile.jks`.
- ▶ The log file under the logs directory is available for tracing security if necessary. The debugging option is disabled by default in the `sas.client.props` file.

8.5.1 Running the WebbankThinClient sample

The client can be executed with two different sets of parameters:

1. Running without programmatic login, only one parameter is required: the WebSphere server's address, for example `was01.itso.ral.ibm.com:900`:
`webbankthinclient was01.itso.ral.ibm.com manager:900 password`
2. Using client side programmatic login, the following parameters should be defined:
 - a. WebSphere's server address, for example: `was01.itso.ral.ibm.com:900`.
 - b. The user ID, for example: `manager`
 - c. The password, for example: `password``webbankthinclient was01.itso.ral.ibm.com manager password`

The result in each case should be the branch account balance of the user, Sophia, for example:

```
Result: balance: 1000
```

Try to run the application with different user IDs, one that has access to the BranchAccount EJB methods and one that does not, and observe the results.

8.6 Applet clients

Applet clients are similar to both the thin Java applications and J2EE application clients. WebSphere provides a special plug-in for Web browsers to enhance them in order to run J2EE clients.

For more information about the applet clients, refer to the WebSphere Infocenter, found at <http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>.

8.7 Authentication summary

The following diagram is a summary of the authentication mechanism for the different kind of clients (*clients* here is used as the most general term).

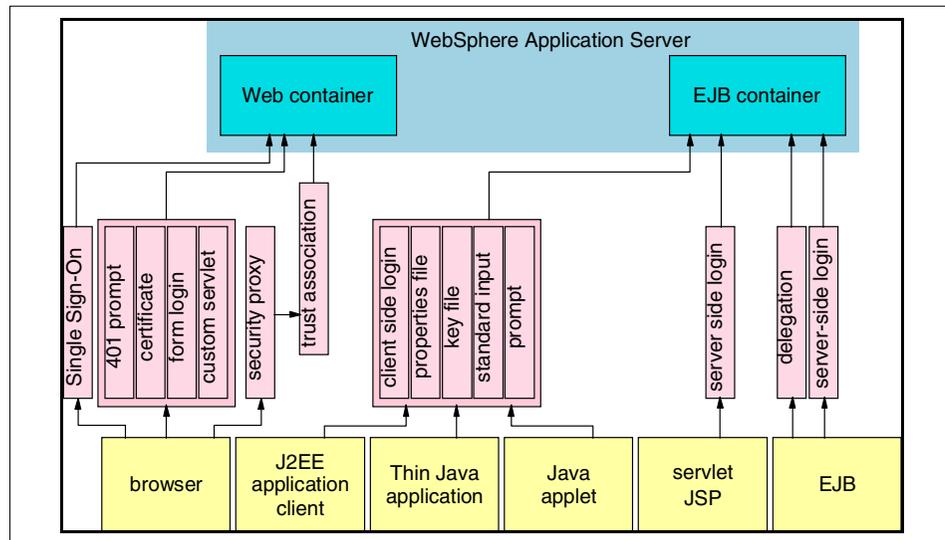


Figure 8-3 Authentication mechanisms overview

At the bottom of this diagram are the different types of clients. These programs can implement EJB clients to access the EJBs.

At the top of the diagram are two containers provided by the application server: the Web container and the EJB container. The clients try to access the assets served by the containers (HTML pages, servlet, JSPs, EJBs).

At the center of the diagram are the authentication mechanisms.

The clients can use authentication mechanisms listed in the center to reach the appropriate container.



Securing Web services

This chapter will show how to secure Web services with WebSphere V4 AE. Web services security is based on messaging security.

Web services are an emerging technology, and companies are starting to use it widely. The need for secure messaging with Web services has grown since Web services have access to sensitive enterprise processes and resources.

This chapter will discuss basic Web services secure messaging, as well as the theory and the practice behind it. It will also show how to create a secure Web service from a session EJB using WebSphere Studio Application Developer.

9.1 Web services

This book does not intend to introduce the concept of Web services. From a security point of view, all that it is necessary to know is that Web services are based on two fundamental technologies: HTTP (the transport protocol for the Web) and XML (the universal markup language).

A correct definition and technical explanation of Web services can be found in the following IBM Redbook: *Web Services Wizardry with WebSphere Studio Application Developer* SG24-6292.

The reader will find that Web services are nothing new from a technical point of view; this technology uses other technologies that have already been implemented for some time. However, it does bring up a new concept of distributed applications and services into the World Wide Web. The underlying technologies are well known, so in order to understand the concept of Web services and to recognize each component discussed here, it is strongly recommended that you read the book mentioned above.

9.2 Securing WebSphere Web services

WebSphere Application Server V4 provides a runtime environment for Web services. Moreover, it provides an early implementation of security for Web services, though this is not yet standardized.

The implementation is based on a W3C document: *SOAP Security Extensions: Digital Signature*, published February 6th, 2001. The document can be found at the following URL: <http://www.w3.org/TR/SOAP-dsig/>. The document proposes a standard way to use the XML Digital Signature Syntax to sign SOAP 1.1 messages.

The XML-Signature Syntax and Processing proposed by W3C on August 20th, 2001 is located at the following URL: <http://www.w3.org/2000/09/xmlsig>.

Numerous other proposals and recommendations have been published and address the security issues left open in the SOAP specification. These documents can also be found at the following URL: <http://www.w3.org/>.

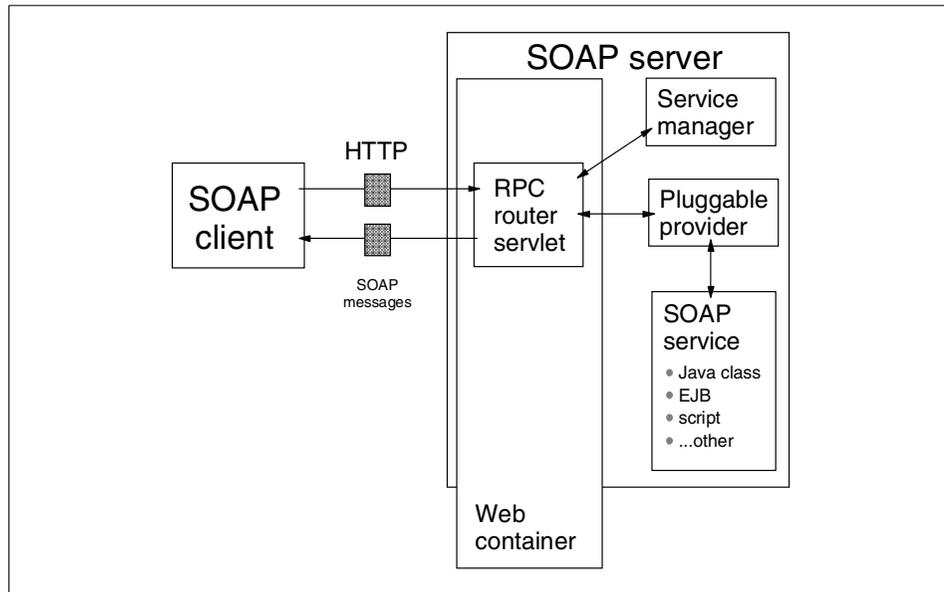


Figure 9-1 Secure Web service component diagram

Figure 9-1 depicts a simple client-server model with the SOAP client and the SOAP server.

The SOAP client sends a request to the RPC router servlet, running within the Web container of the application server, over HTTP.

The RPC router servlet picks up the request, then routes the message to the appropriate pluggable provider, which runs the SOAP service to perform the task.

When the response message is generated, the RPC router servlet sends back the message to the SOAP client.

Please note that the previous example is a simplified realization of Web services. From a security point of view, nothing beyond the SOAP client, the HTTP connection and the RPC router is relevant at this point.

9.2.1 Securing SOAP services

There are three options to secure Web services with WebSphere Application Server:

- ▶ HTTP basic authentication
- ▶ SSL (HTTPS)
- ▶ SOAP signature

These options can be exercised together or can be combined in one implementation.

HTTP basic authentication

The HTTP basic authentication secures access to the Web service itself. Since the Web services are using HTTP as the transfer protocol, the HTTP basic authentication can be applied to the SOAP transfer connection also.

This method is not tightly related to Web services; only the authentication procedure have to be implemented in the SOAP client code.

SSL (HTTPS)

The SSL (HTTPS) secure transfer protocol ensures security during the communication time. It takes advantage of the HTTP and uses the widely accepted SSL to secure the connection between the client and the server.

SOAP signature

SOAP signature secures the message itself with the content inside. Unlike the other two security methods, this is closely related to SOAP messaging.

SOAP signatures will be discussed in detail in the following section.

9.3 SOAP signature components

The SOAP signature is based on the XML signature proposal from W3C. It specifies the procedure of how the SOAP messages should be signed using server and client certificates.

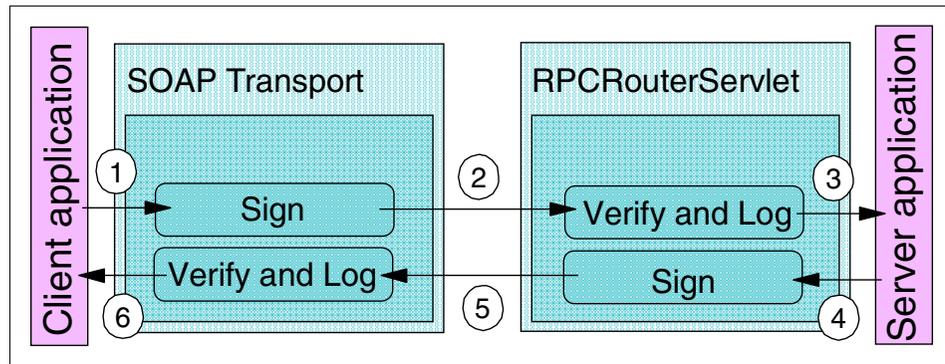


Figure 9-2 Secure transport of SOAP messages

The transport hook is called the *EnvelopeEditor*. The *PluggableEnvelopeEditor* is provided to plug in the security components, as shown in Figure 9-2.

The message from client to server follows this procedure:

1. The client application sends a message.
2. The message is signed, then sent to the RPC Router Servlet.
3. The servlet verifies the message and passes it to the server application. The verifier component also has a logging function to log the verified messages.

The response message from server to client follows this procedure:

4. The server application prepares the response message and sends it back.
5. The message is signed on the server side then sent to the client.
6. The client verifies the message, then processes the response.

9.3.1 Web module

Web services require two servlets at the application server in order to run. These servlets are defined in the Web modules's deployment descriptor, in the web.xml file. The servlet configuration part of the deployment descriptor can be found in Example 9-1 on page 176.

Example 9-1 web.xml

```
...
<servlet>
  <servlet-name>rpcrouter</servlet-name>                                (1)
  <display-name>Apache-SOAP RPC Router</display-name>
  <description>no description</description>
  <servlet-class>
    com.ibm.soap.server.http.WASRPCRouterServlet
  </servlet-class>
  <init-param>
    <param-name>faultListener</param-name>
    <param-value>org.apache.soap.server.DOMFaultListener</param-value>
  </init-param>
</servlet>
<servlet>
  <servlet-name>messagerouter</servlet-name>                            (2)
  <display-name>Apache-SOAP Message Router</display-name>
  <description>no description</description>
  <servlet-class>
    com.ibm.soap.server.http.WASMessageRouterServlet
  </servlet-class>
  <init-param>
    <param-name>faultListener</param-name>
    <param-value>org.apache.soap.server.DOMFaultListener</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>rpcrouter</servlet-name>
  <url-pattern>servlet/rpcrouter</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>messagerouter</servlet-name>
  <url-pattern>servlet/messagerouter</url-pattern>
</servlet-mapping>
...

```

The following two servlets are defined for the Web service under the `webbankWeb` Web module:

1. RPC Router Servlet
2. Message Router Servlet

The servlets are provided by IBM WebSphere Application Server; they are part of the application server's library.

9.3.2 Envelope Editor

The Envelope Editor defines the pluggable security components by specifying the configuration files for the outbound and inbound message handlers.

Example 9-2 shows the Envelope Editor configuration file for the client from the Webbank application.

Example 9-2 cl-editor-config.xml

```
<?xml version="1.0" standalone="yes"?>
<SOAPEnvelopeEditorConfig
  xmlns="http://www.ibm.com/xml/soap/#EnvelopeEditor">
  <incoming class="com.ibm.xml.soap.security.dsig.SOAPVerifier"> (1)
    <init-param>
      <param-name>filename</param-name>
      <param-value>conf/cl-ver-config.xml</param-value>
    </init-param>
  </incoming>
  <outgoing class="com.ibm.xml.soap.security.dsig.SOAPSigner"> (2)
    <init-param>
      <param-name>filename</param-name>
      <param-value>conf/cl-sig-config.xml</param-value>
    </init-param>
  </outgoing>
</SOAPEnvelopeEditorConfig>
```

Two configuration elements have been defined in the envelope editor descriptor.

1. *<incoming>* - this defines the verifier class for the client and the configuration file for verification of the message.
2. *<outgoing>* - this defines the signer class for the client and the configuration file for signing the message.

The server configuration file, sv-editor-config.xml, looks the same, with the exception of the configuration file names.

Removing the digital signature from inbound messages

Since these security components are pluggable, it is possible to remove security from either the inbound or the outbound message flow.

In order to remove the digital signature from the response, remove:

- ▶ The outgoing element from the sv-editor-config.xml file, and
- ▶ The incoming element from the cl-editor-config.xml file.

The server will not sign the response message, and the client will not verify the incoming message from the server.

9.3.3 Signature Header Handler

The Signature Header Handler inserts a digital signature header into a SOAP envelope. The following section will show how to configure the signature for SOAP messages.

Signature component

Example 9-3 is an excerpt from the Webbank application. It shows the client's signature header handler configuration file.

Example 9-3 cl-sig-config.xml

```
<?xml version="1.0"?>
<!DOCTYPE SOAPSignerConfig SYSTEM "sig-config.dtd">
<SOAPSignerConfig
  xmlns="http://www.ibm.com/xml/soap/#SOAPSsignature"
  xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12">
  <KeyStore (1)
    type="jks"
    path="key\SOAPclient"
    storepass="client"
  />
  <Policy> (2)
    <PrivateKey alias="soaprequester" keypass="client"/> (3)
    <PublicKey> (4)
      <IncludeKeyName flag="yes"/>
      <IncludeKeyValue flag="yes"/>
      <IncludeX509Data flag="yes"/>
    </PublicKey>
    <Template> (5)
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo Id="sig">
          <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
          <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <Reference URI="">
            <Transforms>
              <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            </Transforms>
            <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue></DigestValue>
          </Reference>
          <Reference URI="#timestamp"
Type="http://www.w3.org/2000/09/xmldsig#SignatureProperty">
            <Transforms>
```

```

        <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue></DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue></SignatureValue>
    <Object> (6)
        <SignatureProperties>
            <SignatureProperty SOAP-SEC:id="timestamp" Target="#sig">
                <ValueOfTimestamp
xmlns="http://www.ibm.com/xml/soap/#SOAPSignature"/>
                </SignatureProperty>
            </SignatureProperties>
        </Object>
    </Signature>
</Template>
</Policy>
</SOAPSignerConfig>

```

The following important elements are defined in the configuration file:

1. *<KeyStore>* - this tag defines the keystore file, which holds the signing key for the message. The *jks* indicates that this is a Java Key Store type of keystore. You can manage these keystores with the IBM Key Management Tool (iKeyman). The path refers to the *SOAPclient* file, which is created for the secure Web service. The same *SOAPclient* keystore is provided with every secured Web service generated with the WebSphere Studio Application Developer; in a production environment, developers are urged to replace it with their own keystores and certificates.
2. *<Policy>* - this defines the policies regarding secure messaging. This tag encapsulates the details of the messaging.
3. *<PrivateKey>* - this tells the application which private key should be used to sign the message on the client side.
4. *<PublicKey>* - this specifies the information that should be in the *<ds:KeyInfo>* element in the verification header handler's configuration file.
5. *<Template>* - this element specifies the details for the XML signature, including:
 - a. Canonicalization algorithms
 - b. Signature algorithms
 - c. Transformation algorithms

- d. Digest algorithms
 - e. The portion of the SOAP envelope to be signed
6. *<Object>* - this specifies additional authentication information. In this example, only a timestamp is defined.

9.3.4 Verification Header Handler

The Verification Header Handler validates a signature header in a SOAP envelope. This section will show how to configure the signature verification for SOAP messages.

Verification component

Example 9-4 shows part of the Webbank application: the configuration file for the client's verification header handler.

Example 9-4 cl-ver-config.xml

```

<?xml version="1.0"?>
<!DOCTYPE SOAPVerifierConfig SYSTEM "ver-config.dtd">
<SOAPVerifierConfig
  xmlns="http://www.ibm.com/xml/soap/#SOAPSignature"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12">
  <AllowedAlgorithms>
    <Algorithm type="DigestMethod"
      URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <Algorithm type="Encoding"
      URI="http://www.w3.org/2000/09/xmldsig#base64"/>
    <Algorithm type="SignatureMethod"
      URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Algorithm type="CanonicalizationMethod"
      URI="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
    <Algorithm type="Transform"
      URI="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
  </AllowedAlgorithms>
  <RequiredAuthenticatedParts>
    <Reference part="root"/>
  </RequiredAuthenticatedParts>
  <DefaultVerificationKeys>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
MIIDFjCCAn+gAwIBAgICAQEwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCS1AxETAPBgNVBAGT
CEthbmFnYXdhMQwwCgYDVQQKEwNJK0xDDAKBgNVBAsTA1RSTDEQMA4GA1UEAxMH5W50IENBMjAe
Fw0wMTAyMTUwNzAxNTNaFw0wMjAyMTUwNzAxNTNaMFmxCzAJBgNVBAYTAkpQMREwDwYDVQQIEWhL
YW5hZ2F3YTEEMMAoGA1UEChMDSUJNMQwwCgYDVQQLEwNUUkwxFTATBgNVBAMTDFNPQVBQcm92aWR1
cjCBnJANBgkqhkiG9w0BAQEFAAOBjAAwYgCgYBxvYvh1JY9RxxMMia1781jNkKtSrzzxCKcSw5JW

```

(1)

(2)

(3)

```

nK32dxYfL9WITTAoFOyG2Dko0QCnWVqoJ301AYP/WgteQmmSZ0gYyJeVc/GBykBwi7NBENs+pv8q
5ogEXVSFfrN4wyYIkHBCyKbs9J/8tvM8dR1NLcGYI01vT1Zdp1UF70BWoQIDAQABo4H+MIH7MAkG
A1UdEwQMAAwCwYDVROpBAQDAgXgMCwGcWGSAGG+EIBDQqFfH1PcGVuU1NMIEd1bmVYXR1ZCBd
ZXJOaWZpY2FOZTAdBgNVHQ4EFgQU8VbxAKrtbZmXBP0Z2iByIMvkbugwZMGA1UdIwSbiczBiIAU
PUXs0arK5B2FhajR4+jc0B4NA1GhbKRqMGgxCzAJBgNVBAYTAKpQMRewDwYDVQIEwhLYW5hZ2F3
YTEPMAOGA1UEBxMGWwFtYXRvMQwwCgYDVQQKEwNJK0xDDAKBgNVBAsTA1RSTDEZMbcGA1UEAxMQ
U09BUCAyLjEgVGVzdCBDQYICAEwDQYJKoZIhvcNAQEFBQADgYEAq1M4JoOPT17ME8mzoQ7IxEKI
C4GjERCNw14JPXMaZu3emI1kpKhKy0Z27PdTg9xAfPZS7Mk7Kdj6rNhGVAW6CjBx8tMSvsuGIyR
JxG2Wo8Wwexc6nDT+gynkMGEVWOXuKT0ssWtsmyRqVJGD2ZkrD1muepJ7US2piqNmMp2Hs=
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</DefaultVerificationKeys>
<Log> (4)
  <SOAPDSigLogger
    class="com.ibm.xml.soap.security.dsig.SOAPDSigLoggerImpl">
    <LogFile target="all" path="log/SOAPVHH-all-cl.log" append="yes"/>
  </SOAPDSigLogger>
  <SOAPDSigLogger
    class="com.ibm.xml.soap.security.dsig.SOAPDSigLoggerImpl">
    <LogFile target="fail" path="log/SOAPVHH-fail-cl.log" append="yes"/>
  </SOAPDSigLogger>
</Log>
<PKIXParameters> (5)
  <TrustedRootList>
    <TrustedRoot>
      <KeyStore
        type="jks"
        path="key\SOAPclient"
        storepass="client"/>
    </TrustedRoot>
  </TrustedRootList>
  <InitialPolicies>
    <CertificatePolicy>2.4.1.2.3</CertificatePolicy>
    <CertificatePolicy>2.5.1.2.3</CertificatePolicy>
    <CertificatePolicy>2.5.1.7.3</CertificatePolicy>
  </InitialPolicies>
  <PolicyMappingInhibited flag="no"/>
  <ExplicitPolicyRequired flag="no"/>
  <RevocationEnabled flag="no"/>
  <TargetKeyUsage>
    <KeyUsage>DIGITAL_SIGNATURE</KeyUsage>
  </TargetKeyUsage>
  <TargetExtendedKeyUsage>
    <ExtendedKeyUsage>EKU_TIME_STAMPING</ExtendedKeyUsage>
  </TargetExtendedKeyUsage>
  <CertStoreList>
    <LDAPCertStore provider="IBMCertPath">
      <LDAPServer host="localhost" port="389"/>
    </LDAPCertStore>
  </CertStoreList>

```

```
        </LDAPCertStore>
    </CertStoreList>
</PKIXParameters>
</SOAPVerifierConfig>
```

The following important elements are defined in the configuration file:

1. *<AllowedAlgorithms>* - all the algorithms supported by the verification header handler must be listed in this element. It should be consistent to some degree with the algorithms specified in the signature header handler's *<Template>* element.
2. *<RequiredAuthenticatedParts>* - this specifies which parts of the SOAP envelope need to be authenticated through the SOAP-SEC:Signature header. Currently, two values are supported:
 - root - the whole envelope must be signed.
 - body - the SOAP-ENV:Body element in the SOAP envelope must be referenced in the signature; that element must then be signed.
3. *<DefaultVerificationKeys>* - when *<KeyInfo>* is missing in the signature, then the information in this element will be used to perform the verification. It is useful when the communicating parties know each other, because they do not have to exchange certificates; the communication data volume can therefore be reduced.
4. *<Log>* - this specifies the logging behavior; the following settings are available:
 - target="all" - all verification attempts are logged.
 - target="success" - only successful verification are logged.
 - target="fail" - only unsuccessful verification are logged.The logging Java class and the log file are also defined for each logging facility.
5. *<PKIXParameters>* - the policies for PKIX certificate verification are specified in this element. Since the digital signatures for SOAP messages are an early implementation in WebSphere Application Server V4 AE, not all of the entries are meaningful in this release. Currently, the Verification Handler supports X.509/PKIX certificates only.

9.4 How to create secure Web services with WebSphere Studio Application Developer

The Webbank Web application will be enhanced with a new service to provide information about the accounts through a Web service.

In this book, the same Webbank application was used as in the *WebSphere V4 Advanced Edition Handbook* (SG24-6176).

9.4.1 Modifying the Webbank code

The original Webbank code requires some modification in order to be enabled for Web services.

The EJB key classes do not implement the *setter* and *getter* methods for the primary key fields. The bean serializer class for SOAP messaging requires these methods to pass the object as a parameter.

The *BranchAccountKey* and *CustomerAccountKey* objects need to be modified as documented below.

Add the following two methods to the *BranchAccountKey.java* (see Example 9-5).

Example 9-5 Additional code for the BranchAccountKey.java

```
...
public void setBranchID(String argBranchID) {
    branchID = argBranchID; }
public String getBranchID() {
    return branchID; }
...
```

Add the following four methods to the *CustomerAccountKey.java* (see Example 9-6).

Example 9-6 Additional code for the CustomerAccountKey.java

```
public void setCustomerID(String argCustomerID) {
    customerID = argCustomerID; }
public String getCustomerID() {
    return customerID; }
```

```
public void setAccountNumber(String argAccountNumber) {
    accountNumber = argAccountNumber; }
public String getAccountNumber() {
    return accountNumber; }
```

Now the two key objects are enabled to access the primary key field identifiers.

Note: if you do not modify the code and try to use the objects as parameters for messaging, the Web service will fail. The server will get the object as a parameter; it will not be *null*, but the fields will be empty.

9.4.2 Creating the secure Web service

In this section, a new secure Web service will be added to the Webbank Web application; for development, WebSphere Studio Application Developer (WSAD) is used.

1. Switch to the J2EE perspective in WebSphere Studio Application Developer (see the icon on the left side of Figure 9-3).
2. Switch to the J2EE view (see the tab at the bottom of Figure 9-3).
3. Select the **Information** bean under the Consultation bean (**EJB Modules > webbankEJBs > Consultation** bean).

Developers can also select the item to enable Web service through the Web service wizard. By selecting the artifact prior to running the wizard, only the steps for selection will be skipped.

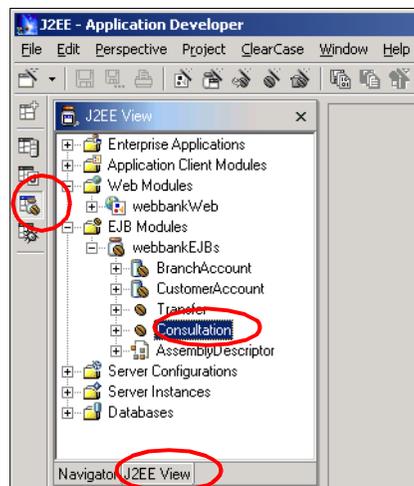


Figure 9-3 Consultation EJB

4. Select **File > New > Other** from the menu to create a new Web service in WebSphere Studio Application Developer.

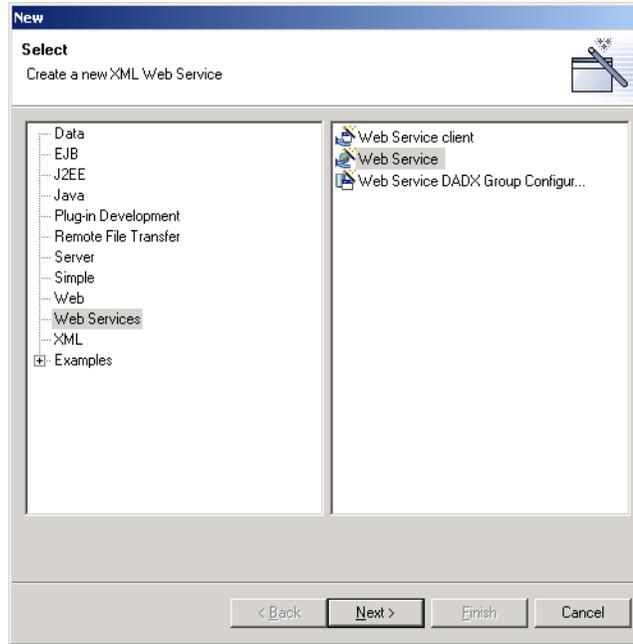


Figure 9-4 Creating a new Web service

5. Select **Web services** in the left-hand pane, then select **Web service** in the right-hand pane. Click **Next** to start the Web services wizard.

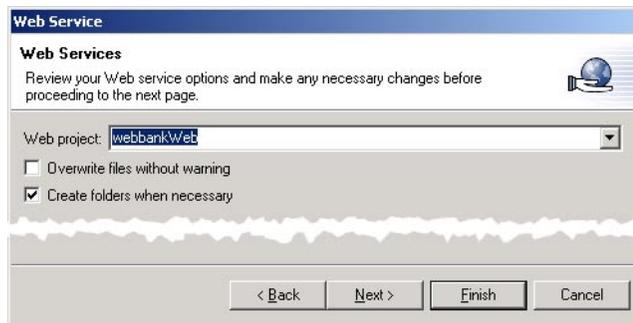


Figure 9-5 Selecting the Web project for the Web service

6. The first window will ask for the Web project where the Web service will be implemented. Select **webbankWeb** from the drop-down menu, as shown in Figure 9-5.

7. Leave the check boxes as they are on the picture, then click **Next**.

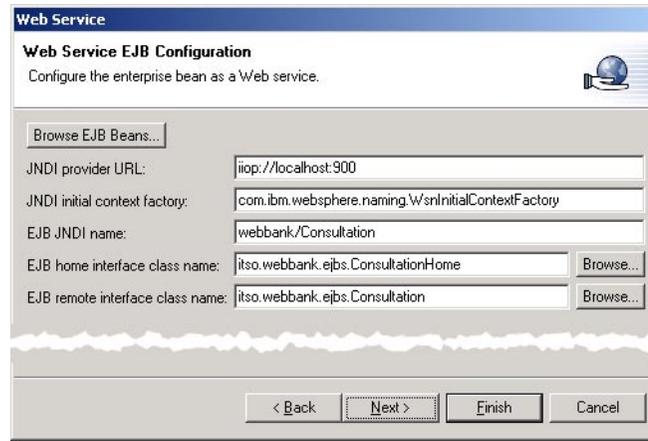


Figure 9-6 Selecting the EJB for the Web service

8. The selected EJB (Consultation) pops up in the Web Service EJB Configuration window, as shown in Figure 9-6. This is why we selected the EJB before we started the wizard. Make sure that all the fields are filled out with the appropriate value, then click **Next**.

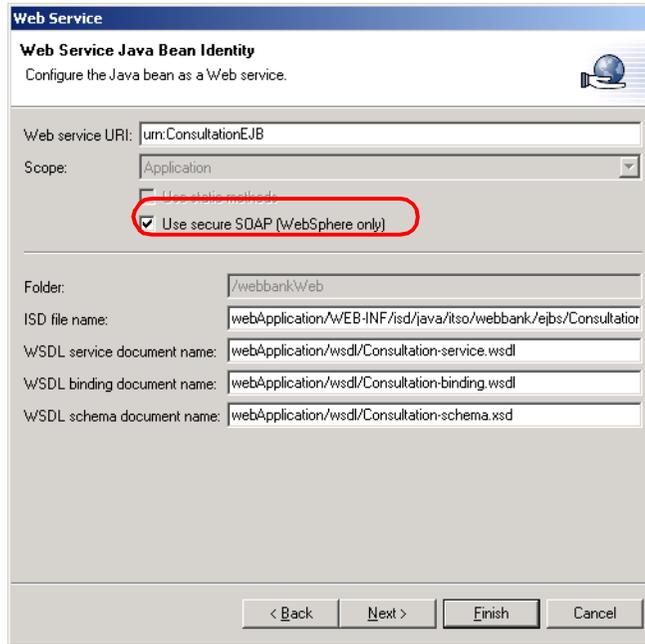


Figure 9-7 Java Bean Identity

9. The Web Service Java Bean Identity window provides the files for the necessary descriptors, such as the ISD and WSDL files. This window also contains the *Use secure SOAP (WebSphere only)* checkbox, which is very important from a security point of view.
10. Make sure that the **Use secure SOAP (WebSphere only)** checkbox is selected (see Figure 9-7), then click **Next..**

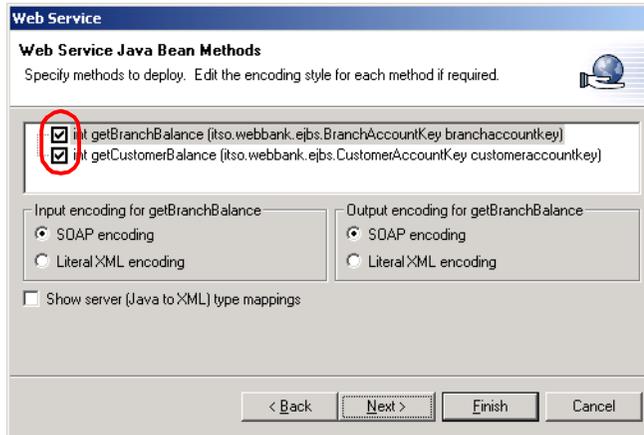


Figure 9-8 Exposed Java Bean methods

11. The Web Service Java Bean Methods window shown in Figure 9-8 lists all the accessible methods from the remote interface of the Consultation EJB. Here, the developers can select those methods which they want to expose for the Web service.

Make sure that the necessary methods are selected, then click **Next**.

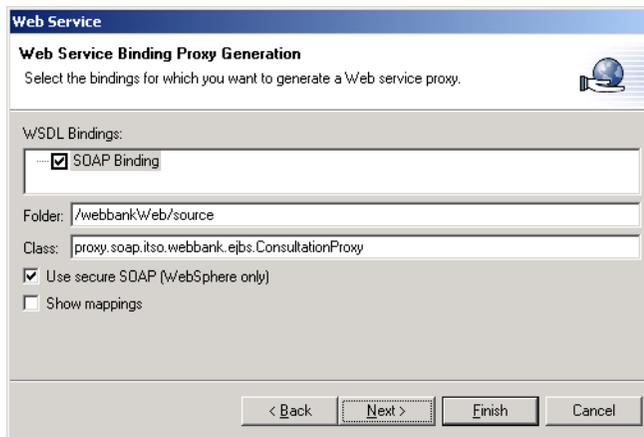


Figure 9-9 Binding Proxy Generation window

12. Figure 9-9 shows the Web Service Binding Proxy Generation window; make sure that **SOAP Binding** and **Use secure SOAP (WebSphere Only)** are selected, then click **Next**.

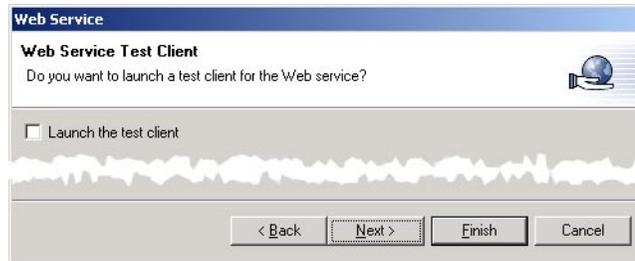


Figure 9-10 Launch Test client

13. The Web Service Test Client window shown in Figure 9-10 asks whether the developer wants to launch the Test client for the Web service after the Web service is generated. Leave the box unselected, then click **Next**.

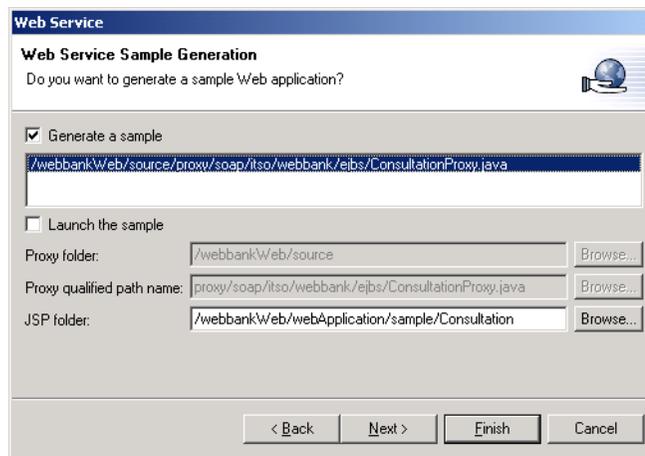


Figure 9-11 Generating the sample

14. The next window, shown in Figure 9-11, will help to generate a sample for the Web service. Select the **Generate sample** checkbox to generate the sample code automatically, then click **Next**.

Generating a sample is a good way to test the Web service either in the development environment or in the runtime environment.

The sample code is also useful for developing the client code. The sample includes all the necessary code that is required for a SOAP client. It provides a good skeleton for coding.

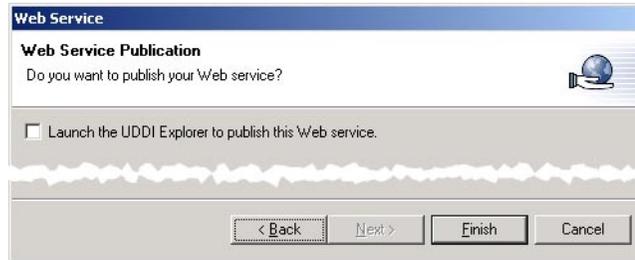


Figure 9-12 Web service publication

15. The Web Service Publication window is the last window of the wizard. Since we will not publish this Web service (the Web service will be accessed directly, so there is no need to publish it), leave the **Launch the UDDI Explorer to publish this Web service** checkbox unselected, then click **Finish**.

After completing the last window of the wizard, it will take a couple of minutes for WebSphere Studio Application Developer to generate all the code. Once it has finished, the WebSphere Test Environment in WebSphere Studio Application Developer will start automatically.

Important: there is a small typographical error in one of the files generated for the secured Web services.

Open the file *conf/ver-config.dtd* and search for the following tag:

```
<!ATTLIST RetrievalMethod
  URI CDATA #REQUIRED
  Type CDATA #IMPLIED >
```

Then add the ds namespace to the RetrievalMethod to get the following:

```
<!ATTLIST ds:RetrievalMethod
  URI CDATA #REQUIRED
  Type CDATA #IMPLIED >
```

Save the file, then close it.

9.4.3 Testing the Web service

After generating the code, you may want to test the code.

1. Open the Server perspective or switch to it if you have it already opened.
2. Find the following Consultation folder by clicking **webbankWeb -> webApplication -> sample -> Consultation**.

3. Right-click the **TestClient.jsp** file, then select **Run on Server**. A Web browser opens with the requested URL.

A window appears with three frames. In the leftmost frame, the following methods should be listed:

- setEndPoint
 - getEndPoint
 - getEnvelopeEditorConfigURL
 - getEnvelopeEditorConfigURL
 - setEnvelopeEditorHomeDirectory
 - getEnvelopeEditorHomeDirectory
 - getBranchBalance
 - getCustomerBalance
4. Click the **getBranchBalance** link.
 5. The upper-right frame changes: a text field appears for you to input the branch ID. Type *Sophia* into the field, then click **Invoke**.
 6. After invocation, the result frame changes and, after a couple of seconds and several messages on the console, the result should appear in the frame.
 7. Check the `getCustomerBalance` method also, with the following parameters:
 - Customer ID: *Isabelle*
 - Account number: *A1*

If the code is working, you should see the proper numbers under the result frame after invoking the methods with the parameters.

9.4.4 Generated code

After generating the code for the Web service, the following files will appear in the project:

Table 9-1 Files created for the secure Web service

File name	Purpose
admin/*	Administering the Web services
conf/sig-config.dtd	DTD for the signature header handler
conf/ver-config.dtd	DTD for the verification header handler
conf/cl-editor-config.xml	Client's envelope editor descriptor
conf/cl-sig-config.xml	Client's signature header handler

File name	Purpose
conf/cl-ver-config.xml	Client's verification header handler
conf/sv-editor-config.xml	Server's envelope editor
conf/sv-sig-config.xml	Server's signature header handler
conf/sv-ver-config.xml	Server's verification header handler
key/SOAPclient	Client's keyring file in JKS
key/SOAPserver	Server's keyring file in JKS
key/sslserver.p12	Keyring file for the SSL connection in PKCS12 format
log/dummy.log	Dummy file in the log directory
sample/Consultation/*	Sample application for testing purposes
wsdl/Consultation-binding.wsdl	WSDL binding configuration file
wsdl/Consultation-service.wsdl	WSDL binding service file
dds.xml	Deployment descriptor for all the services
soap.xml	SOAP server configuration file
WEB-INF/classes/proxy/soap/itso/webbank/ejbs/ConsultationProxy.class	Servlet for the sample application
WEB-INF/isd/java/itso/webbank/ejbs/Consultation.isd	Deployment descriptor for the particular service
WEB-INF/lib/*	Java libraries required for Web services

For more information about Web services and developing Web services, read the IBM Redbook *Web Services Wizardry with WebSphere Studio Application Developer*, SG24-6292.

9.4.5 The XML-SOAP Admin tool

The Web services Admin tool is provided for the sample Web services and is generated automatically with the code. Start a browser, then enter the following URL:

`http://<your_server>/webbank/admin/index.html`

Click the **list all services** link to see all the available services for this sample; you should see the picture shown in Figure 9-13 on page 193.



Figure 9-13 XML-SOAP Admin

The service listing shows one service available: urn:ConsultationEJB.

The services can be stopped or started under the related links within the XML-SOAP Admin tool.

9.4.6 Running the Webbank Web services sample

A Java client is provided with the webbank6520.ear sample together with this redbook.

The syntax for running the sample is:

```
consultation.bat RPC_URL web_application_path [ branch branchID | customer
customerID account_number ]
```

There are two options: requesting information about the branch or about the customer, depending on the parameters passed to the Java application.

After running the application, the following output is generated in the console.

Example 9-7 Console output

```
[Monday, November 12, 2001 10:39:10 AM EST] main "EditorComponent class name: c
om.ibm.xml.soap.security.dsig.SOAPVerifier"
[Monday, November 12, 2001 10:39:10 AM EST] main "{EnvelopeEditorHome=e:/projec
ts/webbankd/webbankWeb/webApplication, filename=conf/cl-ver-config.xml}"
```

```

[Monday, November 12, 2001 10:39:10 AM EST] main "conf/cl-ver-config.xml"
[Monday, November 12, 2001 10:39:14 AM EST] main "EditorComponent class name: com.ibm.xml.soap.security.dsig.SOAPSigner"
[Monday, November 12, 2001 10:39:14 AM EST] main "{EnvelopeEditorHome=e:/projects/webbankd/webbankWeb/webApplication, filename=conf/cl-sig-config.xml}"
[Monday, November 12, 2001 10:39:14 AM EST] main "conf/cl-sig-config.xml"
[Monday, November 12, 2001 10:39:29 AM EST] main "Core validity=true"
[Monday, November 12, 2001 10:39:29 AM EST] main "Signed info validity=true"
[Monday, November 12, 2001 10:39:29 AM EST] main "Signed info message=null"
[Monday, November 12, 2001 10:39:29 AM EST] main "Ref[0](validity=true, message=Ok., uri=, type=)"
[Monday, November 12, 2001 10:39:29 AM EST] main "Ref[1](validity=true, message=Ok., uri=#timestamp, type=http://www.w3.org/2000/09/xmlsig#SignatureProperty)"

customer balance:900

```

You will see a message generated by the SOAP library, and the result at the end of the output.

9.4.7 Checking the log file

At the same time, logs are generated about the secure messaging. The log files can be found at the following location: <WebSphere install directory>\installedApps\webbank6520.ear\webbankWeb.war\log. The following log files are generated:

- ▶ SOAPVHH-all-cl.log
- ▶ SOAPVHH-all-sv.log
- ▶ SOAPVHH-fail-cl.log
- ▶ SOAPVHH-fail-sv.log

These files are specified in the verification header handler configuration files. The logs with the tag *all* in the middle contain all the logs for verifying the messages, while the logs with the *fail* tag in the middle only contain the errors for unsuccessful message verifications.

The following example is an excerpt from the SOAPVHH-all-sv.log, after requesting information about a customer account.

Example 9-8 SOAPVHH-all-sv.log entry

```

Friday, November 9, 2001 12:39:16 PM EST "SUCCESS" <?xml version="1.0"
encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header>

```



```

YW5hZ2F3YTEMMAoGA1UEChMDSUJNMQwwCgYDVQQLEWNUUkwxFjAUBGNVBAWTDVNPQVBSZXF1ZXNO
ZXIwZ3wDQYJKoZIhvcNAQEBBQADgYOAAMIGJAoGBAM47KaB0t0MsCKXsDXWyzM5KGnur0RJPtArO
nIXsvjGrwXuuFeAWVrehvzDimrhOUWmOLTAOLz7q4+W+dCrFDrcC2icWD2TmaisgPMNf8aYjBmwr
DCpKKfHybApjKV904sYjcp4518B31z0SuV0RzmYp80MLVTOs+ACyDunF7IJHAgMBAAGjg4wgfsw
CQYDVROTBAlwADALBgNVHQ8EBAMCBeAwLAYJYIZIAyb4QgENBB8WHU9wZW5TU0wgR2VuZXJhdGVk
IEN1cnRpZmljYXR1MBOGA1UdDgQWBBSsFkUm1rJnoRNjVzSE5saqj3UbPTCBkwYDVROjBGLMIGI
gBQ9Rew5qsrkHYWFqNHj6NzQHgOCUaFspGowaDELMakGA1UEBhMCS1AxETAPBgNVBAGTCEthbFmFn
YXdhMQ8wDQYDVQQHEWZyYW1hdG8xDDAKBgNVBAoTA01CTEMMAoGA1UECXMDFJMMRkwFwYDVQQD
ExBTT0FQIDUmsBUZXR1EiENBggIBATANBgkqhkiG9w0BAQUFAA0BgQA5KACyCidMn4VRuUNAsNOH
Vn7D9JLDY3Y+Knf7V/EE0IGhryuLf+pwvnxibyKZX1C2mwZVVYE/1Mhw1IWDku56VqUh/XGfbDUe
01ZYYDqRhW60p7ghmmnAINmZ6q725xjHwN01JdcZRJdgZyk6io6/Vs1IXfayVh1+8omY9f8Shw==
</X509Certificate>
  </X509Data>
</KeyInfo>
<Object>
  <SignatureProperties>
    <SignatureProperty SOAP-SEC:id="timestamp" Target="#sig">
      <timestamp>Fri Nov 09 12:39:15 EST 2001</timestamp>
    </SignatureProperty>
  </SignatureProperties>
</Object>
</Signature></SOAP-SEC:Signature>
</SOAP-ENV:Header>

<SOAP-ENV:Body>
  <ns1:getCustomerBalance
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:ConsultationEJB">
    <customeraccountkey
xmlns:ns2="http://www.consultation.com/schemas/ConsultationRemoteInterface"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:itso.webbank.ejbs.CustomerAccountKey">
      <customerID xsi:type="xsd:string">Isabelle</customerID>
      <accountNumber xsi:type="xsd:string">A1</accountNumber>
    </customeraccountkey>
  </ns1:getCustomerBalance>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

(5)

(6)

The following important elements are defined in the configuration file:

1. `<Reference URI="">` - this includes the digest value for the message, using the transformation algorithm specified within this element.
2. `<Reference URI="#timestamp">` - this includes the generated digest value for the timestamp, using the transformation algorithm specified within this element.
3. `<SignatureValue>` - this contains the signature value for the message.
4. `<KeyInfo>` - this includes the values for the key: key name, RSA key modulus and the certificate.
5. `<timestamp>` - the time when the message was transmitted.
6. `<SOAP-ENV:Body>` - the message body with the message itself.

The log in the `SOAPVHH-all-cl.log` file, which contains the logs for the client, is very similar to the one previously introduced.

9.5 Customizing the certificates for secure Web services

In a production environment, it is strongly recommended that you use your own certificates: certificates from a trusted certificate authority or self-signed certificates based on the nature of the business where Web services are exercised.

The certificates provided with the WebSphere Application Server or the WebSphere Studio Application Developer are useful for development and testing purposes, but not for production.

9.5.1 Certificates provided by WebSphere Studio Application Developer

The original certificates provided with the generated secure Web service are depicted in Figure 9-14.

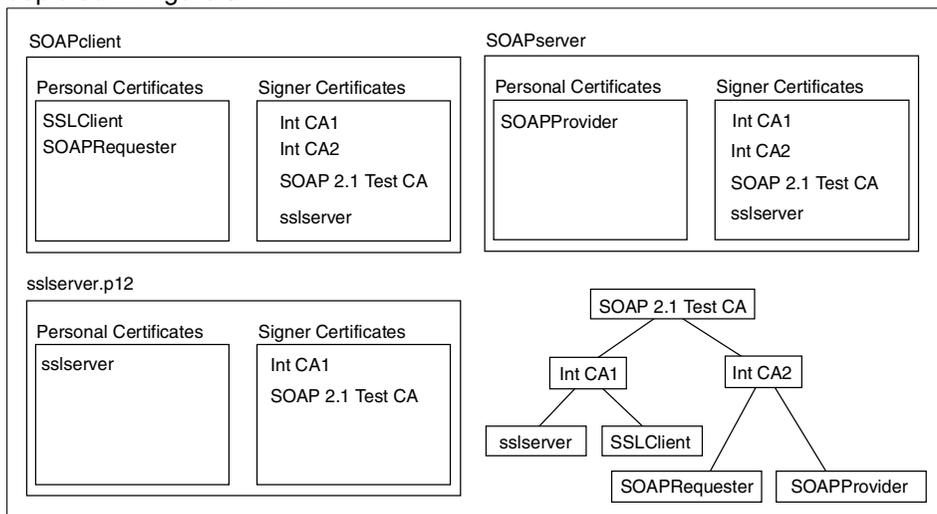


Figure 9-14 Certificates provided with the generated Web service

In Figure 9-14, three different files with certificates are shown. Each keystore contains a group of personal certificates and a group of signer certificates.

The tree shows the hierarchy of the certificates according to the certificate signers. The top level certificate (SOAP 2.1 Test CA) is the root certificate signed by itself. The other certificates are signed by the ones above them.

Note: The certificates provided with the WebSphere secure Web services samples are the same as those provided with WebSphere Studio Application Developer.

9.6 Secure Web services samples in WebSphere V4 AE

WebSphere Application Server V4 Advanced Edition is shipped with a sample enterprise application which demonstrates Web services. The sample includes the secure and the non-secure Web services in one archive, under separate Web archives.

The .ear file is in the <WebSphere install directory>/installableApps directory called soapsamples.ear.

For more information about deploying an enterprise application, refer to the *WebSphere V4 Handbook*, SG24-6176.



Programmatic security

In this chapter, we will look at programmatic security. First, we will establish when the security infrastructure provided by WebSphere may need to be enhanced. Then we will look at three ways in which you can implement programmatic security:

- ▶ Security in the J2EE APIs
- ▶ CustomRegistry SPI
- ▶ TrustAssociationInterceptor SPI

10.1 Programmatic security

WebSphere is able to provide method level security. That is, you can control who can call which methods using the WebSphere administration tools. However, this is not always enough. For example, with our Webbank application, we configured security so that only managers and employees could transfer funds, but anyone could check their balance. However, there is no check performed to ensure that the balance that is being requested actually is for an account that belongs to the user.

This scenario requires a finer grained level of security and that is provided by the WebSphere security infrastructure. You need row-level security, that is, you need to check that the user is allowed to access the data in a particular row of the database. This check can be performed at different points.

If the only purpose of logging into your Web site is to get personal information, then it makes sense to perform this check during the login stage.

If your Web site offers personal information but most of the information is general, you might want to perform this check just in time for the personal information to be presented. In this case, before the `getBalance` method was called, you would validate that the user was trying to access his or her own account. This would be a valid scenario for our Webbank application, since the account of which to display the balance is shown after login.

There are other reasons why you might want to use programmatic security, such as:

- ▶ Limiting the number of invalid password attempts
- ▶ Checking that the user's subscription has not expired
- ▶ Logging information about a user's visit

Programmatic security puts a burden on your developers. This is why WebSphere security provides a security infrastructure, thanks to which the programmer does not have to be concerned with security. However, if WebSphere does not provide everything that you need for your Web site, you do have the ability to code for this. This has an effect on your project, as your code needs to be modified and the security code maintained. Always separate the security logic from your business logic to make it easy to maintain and to give you the flexibility to plug in other security products and features where available.

10.2 J2EE API

WebSphere provides a security infrastructure for application security which is transparent to the application developer. That is, the developer does not need to code for security, since it will all be handled at deployment and runtime.

Having said that, when developing servlets and EJBs, there are a few security calls available if the developer wants greater control of what the end user is allowed to do than is provided by the infrastructure.

10.2.1 EJB security methods

The EJB 1.1 specification defines two methods that allow programmatic access to the caller's security context, *javax.ejb.EJBContext*.

► **java.security.Principal getCallerPrincipal()**

The *getCallerPrincipal* method allows the developer to get the name of the current caller. To do this, you need to call *getName()* on the *java.security.Principal* object returned.

```
EJBContext ejbContext;
...
// get the caller principal
java.security.Principal callerPrincipal =
    ejbContext.getCallerPrincipal();
// get the caller's name
String callerName = callerPrincipal.getName();
```

The *Principal.getName()* method returns the login name of the user.

► **Boolean isCallerInRole(String roleName)**

The *isCallerInRole* method allows the developer to make additional checks on the authorization rights of a user which are not possible, or more difficult, to perform through the deployment descriptor of the EJB.

```
EJBContext ejbContext;
...
if (ejbContext.isCallerInRole("Supervisor"))
    // Perform some fuction
else
    // Throw a security exception
```

The *isCallerInRole(String role)* method returns *true* if the user is in the specified role, and *false* if it is not. The role name specified in the method is really a security role reference, not a role. If the security role reference is not defined for the EJB, the method will return *null*.

See the EJB 1.1 specification for more details on EJB programmatic security.

A typical use of programmatic security with EJBs is logging. If security logs are required in the applications, requiring information like: *Who? When? and Which method was running?* then the logging facility can get the user information and write it out to the log.

Programmatic security sample for EJBs

The *Transfer* session EJB implements a short code, which checks the user's role and will not let an Employee transfer more than \$1000 from the branch to the customer (it will, however, allow the transaction to flow other way around: from the customer to the branch).

Two environment variables are defined for the Transfer session EJB, which are local to this EJB (the namespace where they are defined is not available only from the EJB where they were set). The code checks to see whether the user falls into the role defined in the RestrictedUser environment variable, and will not let the user perform the transfer if the amount is higher than the OverdraftValue environment variable's value.

Since the EJB security API can only check the role mappings through a security role reference, the following entry has to be added to the ejb-jar.xml file, into the body of the <session id="Session_1"> </session> tag.

```
<security-role-ref id="SecurityRoleRef_1">
<role-name>Employee</role-name>
<role-link>Employee</role-link>
</security-role-ref>
```

The security role reference feature is not implemented in WebSphere Studio Application Developer; you must add the entry manually or use the Application Assembly Tool (AAT). For more information on how to define a security role reference for an EJB, see "Setting up security role references" on page 145.

10.2.2 Servlet security methods

The Servlet 2.2 specification defines three methods that allow programmatic access to the caller's security information of HttpServletRequest interface.

► **String getRemoteUser()**

The getRemoteUser method returns the user name that the client used to log in.

```
String user = request.getRemoteUser();
```

► **Boolean `isUserInRole(String roleName)`**

The *isUserInRole* method allows the developer to perform additional checks on the authorization rights of a user which are not possible, or more difficult, to perform through the deployment descriptor of the servlet.

For example, in our Webbank application, we have defined a *Manager* role which can transfer funds of more than \$5000. This is not possible to define using the deployment descriptor alone, so the developer has coded in a check to make sure that the user is a member of the appropriate role.

Important: The methods *getRemoteUser()* and *getUserPrincipal()* return `null` as a result even if the user is logged in, unless the servlet or the JSP itself is secured.

Example 10-1 Programmatic login sample for servlets

```
if (amount > 5000) {
    String message = null;
    if (request.isUserInRole("Manager")) {
        message = TransferHelper.singleton().makeTransfer(
            customerID, accountID, branchID, amount, transferType);
    } else {
        message = "You are not a Manager";
    }
    req.setAttribute("Message", message);
}
```

► **java.security.Principal `getUserPrincipal()`**

The *getUserPrincipal* method allows the developer to get the name of the current caller. To do this, you need to call *getName()* on the `java.security.Principal` object returned.

See the Servlet 2.2 specification for more details on servlet programmatic security.

Programmatic security sample for servlets

The *TransferServlet* uses a simple code which checks for the user role and the transferable amount. If the user falls into the role specified in the Web module's environment (`RestrictedUser`), then the amount of the transfer is compared to another environment variable defined in the Web module's environment (`OverdraftValue`).

The environment variables are local for the Web module. In our example, any user who is in the Employee group cannot transfer more than \$5000 at once. See *TransferServlet.java* for more details.

10.3 CustomRegistry SPI

The custom registry facility in WebSphere allow you to use any system you like to store the users and groups. The only requirement is that there must be a Java interface for it. Some examples of products that could be used as a custom registry are:

- ▶ A database, such as DB2
- ▶ An integration product, such as MQSeries
- ▶ A combination of multiple registries, such as LDAP and RACF

In the case of MQSeries, you would need to implement the CustomRegistry interface and also write an MQ application on the receiving end which interfaced to some back-end repository, such as a mainframe registry.

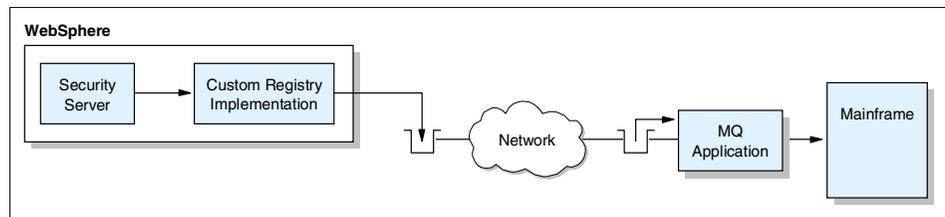


Figure 10-1 MQ Series example

There are three steps to implementing a custom registry:

1. The application developer needs to implement the methods in the CustomRegistry interface, part of the `com.ibm.ejs.security.registry` package. This layer of code interacts with the actual registry.
2. In the Security Center of the WebSphere Administrator's Console, the administrator selects the **Custom pluggable registry** option and specifies the class that was created in step 1. For details on how to configure this, see "User registry" on page 50.
3. The WebSphere Security Server calls the CustomRegistry methods to perform authentication for applications.

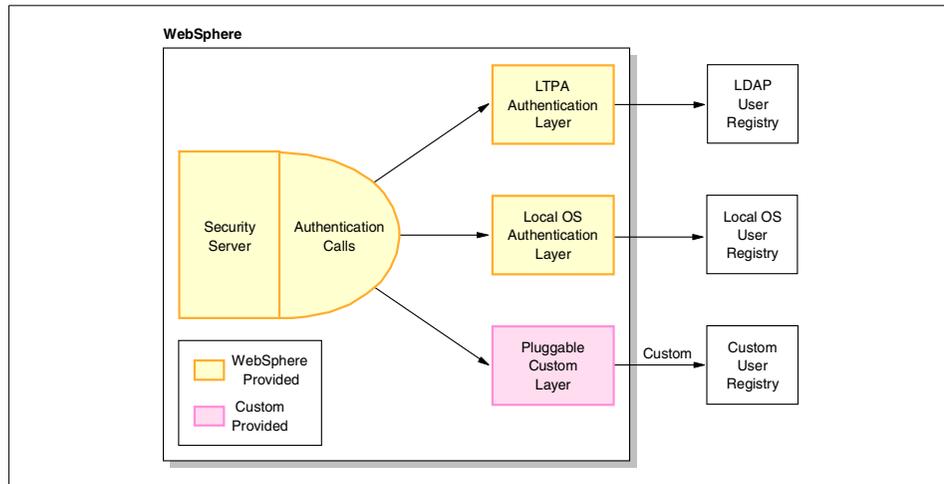


Figure 10-2 Custom Registry

There is an implementation of the CustomRegistry interface shipped with WebSphere. It is a simple file system-based implementation, designed as an example, not to be implemented in a production system. It has not been designed to scale or to perform well.

For more information about this example registry, see Section 5.2 of the InfoCenter. The source code listing for the sample is also in the InfoCenter in Section 5.2.4.1.1 (*The FileRegistrySample.java file*).

The CustomRegistry interface supports two types of registry entry, and three pieces of information for each:

- ▶ Users
 - User name: user identifier
 - Unique identifier: must be unique within the registry
 - Display name: optional description of the user
- ▶ Groups
 - Group name: group identifier
 - Unique identifier: must be unique within the registry
 - Display name: optional description of the group

There are twenty methods in the CustomRegistry interface. An implementation of this interface must include an implementation for all of these methods.

Table 10-1 CustomRegistry required general methods

Method signature (General methods)	Use
Public void initialize(java.util.Properties props) throws CustomRegistryException	Initializes the custom registry.
Public String getRealm() throws CustomRegistryException	Returns the name of the security realm. If null, defaults to customRealm.

The following table lists the user-related methods.

Table 10-2 CustomRegistry required user-related methods

Method signature (user-related methods)	Use
Public boolean isValidUser(String userName) throws CustomRegistryException	Determines if the supplied user name exists in the registry.
Public List getUsers() throws CustomRegistryException	Returns the list of all users in the registry.
Public List getUsers(String pattern) throws CustomRegistryException	Returns the list of all users matching a pattern in the registry. Includes wildcards.
Public String getUniqueUserId(String userName) throws CustomRegistryException, EntryNotFoundException	Returns the unique identifier for the named user.
Public String getUserSecurityName(String uniqueUserId) throws CustomRegistryException, EntryNotFoundException	Returns the user name given the unique identifier.
Public String getUserDisplayName(String securityName) throws CustomRegistryException, EntryNotFoundException	Returns the description for the named user.
Public List getUsersForGroup(String groupName) throws CustomRegistryException, EntryNotFoundException	Returns a list of the users belonging to the named group.
Public List getUniqueUserIds(String uniqueGroupId) throws CustomRegistryException, EntryNotFoundException	Returns all the unique identifiers for users belonging to the named group.

Table 10-3 lists the group-related methods.

Table 10-3 CustomRegistry required group-related methods

Method signature (group-related methods)	Use
public boolean isValidGroup(String groupName) throws CustomRegistryException	Determines if the supplied group name exists in the registry.
Public List getGroups() throws CustomRegistryException	Returns the list of all groups in the registry.
Public List getGroups(String pattern) throws CustomRegistryException	Returns the list of all groups matching a pattern in the registry. Includes wildcards.
Public String getUniqueGroupId(String groupName) throws CustomRegistryException, EntryNotFoundException	Returns the unique identifier for the named group.
Public String getGroupSecurityName(String uniqueGroupId) throws CustomRegistryException, EntryNotFoundException	Returns the group name given the unique identifier.
Public String getGroupDisplayName(String groupName) throws CustomRegistryException, EntryNotFoundException	Returns the description for the named group.
Public List getGroupsForUser(String userName) throws CustomRegistryException, EntryNotFoundException	Returns a list of the groups that the named user belongs to.
Public List getUniqueGroupIds(String uniqueUserId) throws CustomRegistryException, EntryNotFoundException	Returns all the unique identifiers for groups that the named user belongs to.

Table 10-4 shows the CustomRegistry required authentication methods.

Table 10-4 CustomRegistry required authentication methods

Method signature (Authentication methods)	Use
Public String checkPassword(String userId,String password) throws PasswordCheckFailedException, CustomRegistryException	Returns the user name if the password supplied matched the password in the registry entry.
Public String mapCertificate(X509Certificate cert) throws CertificateMapNotSupportedException, CertificateMapFailedException, CustomRegistryException	Takes an X.509 certificate as the argument and returns a valid user.

To illustrate some of the different possible implementations, we have chosen one method from this interface, *getUsers()*, and will show three implementation examples:

- ▶ File-based registry (the implementation supplied in the InfoCenter)
- ▶ Database registry
- ▶ MQSeries (which, in turn, interfaces to a remote registry)

Example 10-2 shows the *getUsers()* method for the Custom Registry using the file for user registry.

Example 10-2 File-based registry implementation of *getUsers()*

```

/* Returns names of all the users in the registry.
 * @return a List of the names of all the users.
 * @exception CustomRegistryException if the registry is "bad".
 */
public List getUsers() throws CustomRegistryException {
    String s;
    BufferedReader in = null;
    List allUsers = new ArrayList();
    try {
        in = fileOpen(USERFILENAME);
        while ((s=in.readLine())!=null) {
            if (!s.startsWith("#")) {
                int index = s.indexOf(":");
                allUsers.add(s.substring(0,index));
            }
        }
    } catch (Exception ex) {
        throw new CustomRegistryException(ex.getMessage());
    } finally {
        fileClose(in);
    }
}

```

```

    }
    return allUsers;
}

```

In Example 10-3, the method implements the *getUsers()* method using a database registry.

Example 10-3 Database registry implementation of getUsers()

```

/* Returns names of all the users in the registry.
 * @return a List of the names of all the users.
 * @exception CustomRegistryException if the registry is "bad".
 **/
public List getUsers() throws CustomRegistryException {
    return internalGetAllIds( SQL_GET_ALL_USERIDS );
}
private List internalGetAllIds( final String statement )
    throws CustomRegistryException {
    IRetryBlock rb = new RetryBlock() {
        public void doRun() throws Exception {
            m_preparedStatement = m_connection.prepareStatement( statement );
            m_resultSet = m_preparedStatement.executeQuery();
            List list = new ArrayList();
            while( m_resultSet.next() )
                list.add( m_resultSet.getString( 1 ) );
            setReturnValue( list );
        }
    }
    m_retryHandler.execute( rb );
    Object rc = rb.getReturnValue();
    if( rb.isSuccessful() == false )
        throw new CustomRegistryException();
    return ( List ) rb.getReturnValue();
}

```

In Example 10-4, the method implements the *getUsers()* method using MQSeries to access the registry.

Example 10-4 MQSeries implementation of getUsers()

```

/* Returns names of all the users in the registry.
 * @return a List of the names of all the users.
 * @exception CustomRegistryException if the registry is "bad".
 **/
public java.util.List getUsers()
    throws com.ibm.websphere.security.CustomRegistryException {
    String inMessage = "getUsers:";
    List replyList = requestReplies(inMessage);
    return replyList;
}

```

```

}
/* Generic method that takes a string and returns a list of strings.
 * @return a List of strings.
 * @exception CustomRegistryException if the registry is "bad".
 **/
private List requestReplies(String name) throws CustomRegistryException {
    Message inMessage = null;
    try {
        boolean transacted = false;
        try {
            // Create a QueueReceiver
            QueueSession session = connection.createQueueSession(transacted,
                Session.AUTO_ACKNOWLEDGE);
            QueueSender queueSender = session.createSender(sendQueue);
            TextMessage jmsMessage = session.createTextMessage();
            jmsMessage.setText(name);
            queueSender.send(jmsMessage);
            //Get the correlation ID of the message
            String jmsCorrelationID = jmsMessage.getJMSCorrelationID();
            String selector = "JMSCorrelationID = '" + jmsCorrelationID + "'";
            // Create a QueueReceiver using the correlation ID
            QueueReceiver queueReceiver = session.createReceiver(receiveQueue,
                selector);
            inMessage = queueReceiver.receive(500);
            queueReceiver.close();
            queueSender.close();
            session.close();
            session = null;
        } catch (JMSEException jmse) {
            jmse.printStackTrace();
        }
    } catch (Exception ex) {
        throw new CustomRegistryException(ex.getMessage());
    }
    // Convert received message into a List object
    List allItems = new ArrayList();
    if (inMessage instanceof TextMessage) {
        String replyString = null;
        try {
            replyString = ((TextMessage) inMessage).getText();
        } catch (JMSEException jmse) {
            jmse.printStackTrace();
        }
    }
    int i = 0;
    int j = replyString.indexOf(":");
    while (j >= 0) {
        allItems.add(replyString.substring(i, j));
        i = j + 1;
        j = replyString.indexOf(":", i);
    }
}

```

```
    }  
    allItems.add(replyString.substring(i));  
    return allItems;  
  } else  
    return null;  
}
```

10.4 Trust Association Interceptor SPI

The *TrustAssociationInterceptor* in WebSphere allows third-party products to provide the authentication services to WebSphere. Reverse Proxy Security Servers (RPSS) can be plugged into WebSphere by implementing this service provider interface.

WebSEAL is an example of an RPSS and WebSphere includes an implementation for this proxy. If you want to use a different RPSS with WebSphere, then this interface needs to be implemented for that specific product. See “Web Trust Association” on page 381 for details on how to configure WebSEAL with WebSphere using Trust Association.

WebSphere provides two ways for you to implement the interceptor class:

► Using the interface

```
com.ibm.websphere.security.TrustAssociationInterceptor
```

This defines three methods that you must implement:

- `public boolean isTargetInterceptor(HttpServletRequest req) throws WebTrustAssociationException;`

This verifies that the proxy making the request is the proxy server that has been configured as the interceptor.

- `public void validateEstablishedTrust(HttpServletRequest req) throws WebTrustAssociationException;`

This determines whether or not the proxy that made the request is trusted. One way to do this is to authenticate the proxy user name and password against the LDAP registry. This is how the WebSEAL interceptor works.

- `public String getAuthenticatedUsername(HttpServletRequest req) throws WebTrustAssociationException;`

This extracts the client’s user name from the HTTP head so that WebSphere can authorize the user for the requested resource.

► Sub-classing the class

`com.ibm.websphere.security.WebSphereBaseTrustAssociationInterceptor`

If you want to be able to configure the interceptor, using a configuration that is read by the class, then you need to use this method. Additionally, you need to implement two further classes:

- `abstract public int init(String propsfile);`
- `abstract public void cleanup();`

The configuration file is specified in the `trustedservers.properties` file with the following syntax:

```
com.ibm.websphere.trustassociation.<proxyname>.config
```

See “Using Tivoli Policy Director” on page 355 for details about the *trustedservers.properties* file used with WebSEAL.

For more information about the `TrustAssociationInterceptor` SPI, see Section 5.6 of the InfoCenter at

<http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>.

For information about programmatic login, refer to “Programmatic login” on page 161.



Administering WebSphere Security

Complementing the *WebSphere Advanced Edition Handbook*, this chapter describes in further detail the steps necessary to configure secure communication between the various components in a WebSphere deployment.

The topics discussed in this chapter include:

- ▶ WebSphere Global Security and Demo Keyring replacement
- ▶ Configuring the Web Server to support HTTPS
- ▶ Client-Side Certificates for authentication
- ▶ Configuring SSL between the Web Server plug-in and WebSphere
- ▶ Restricting Client requests to HTTPS
- ▶ Securing WebSphere LTPA queries with SSL

The documentation is based on WebSphere AE running on the AIX 4.3.3 platform. The certificates were acquired from the Thawte test CA at: <http://www.thawte.com>.

11.1 WebSphere Global Security

Enabling Global Security is the first step in protecting resources served by WebSphere. Fundamentally, this step also secures the WebSphere internal communications mechanism. Here, the Object Request Broker (ORB) requirements associated with the WebSphere Administration Server are endorsed by the Secure Association Service (SAS). This in turn provides the authentication and authorization process, by which client credentials are substantiated.

It is, however, with the adoption of the Secure Socket Layer (SSL) transport mechanism that WebSphere achieves secure communication. As such, the various SSL implementations found in a typical WebSphere environment are illustrated in Figure 11-1 below, and include the HTTPS, LDAPS and IIOP with SSL protocols.

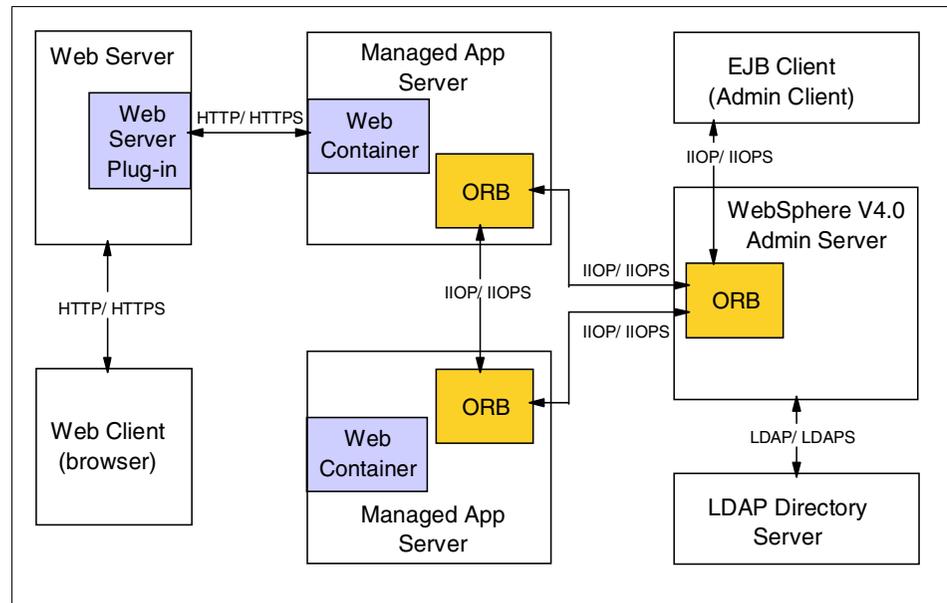


Figure 11-1 Secure protocols adopted in WebSphere V4

One of the features of the Secure Socket Layer (SSL) transport mechanism is the reliance placed upon digital certificates. The use of such certificates permits the authentication and encryption required between peers when establishing a secure connection. By default, WebSphere ships with a set of generic SSL certificates that allow Global Security to be quickly configured and enabled. For this reason, in the section that follows, we explore the tasks and options available for replacing the out-of-the-box SSL certificates.

11.1.1 The Demo Keyring

It is strongly advised that the Demo Keyring be replaced with either a new self-signed certificate key pair or a certificate signed by a third-party Certificate Authority (CA) prior to enabling WebSphere Global Security in a production environment. The CA option, while valid, is perhaps not of any extra benefit as the trust association offered by the CA is of minimal concern to WebSphere internally. We will, however, demonstrate both methods for the sake of completeness.

Failing to replace the Demo Keyring may compromise your overall security integrity, as the same private/public key pair is generically distributed on all shipments of the WebSphere install media. This may prove to be the Achilles' heel of your implementation, if you have spent a considerable amount of time configuring the other elements of WebSphere security.

Prior to WebSphere V4, the Demo Keyring was shipped as a single Java class: the DummyKeyring.class. Now, the Demo Keyring is referenced by WebSphere by its inclusion in the DummyServerKeyFile.jks Java Key Store (JKS). For migration and recovery purposes, the Demo Keyring is also included in the DummyKeyring.jks file. Both files are found in the WebSphere *etc* directory.

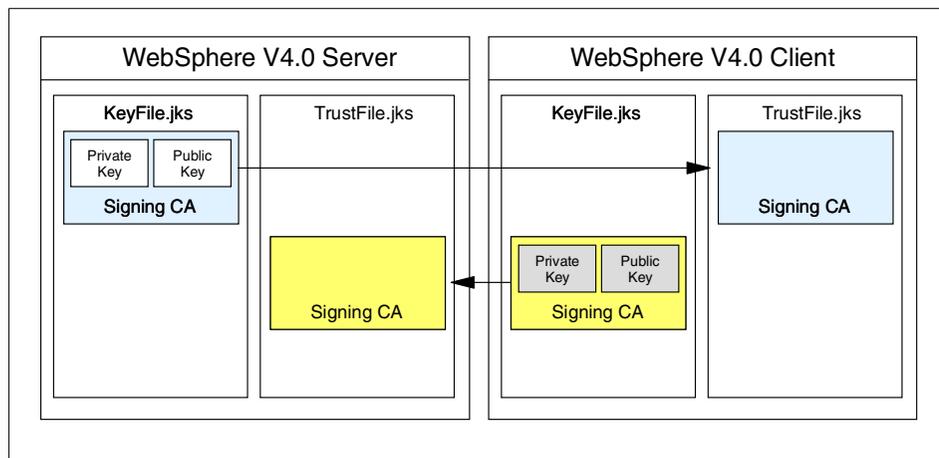


Figure 11-2 JSSE KeyFile.jks and TrustFile.jks relationship

With WebSphere adopting the Java Secure Sockets Extension (JSSE) standard, the concept of separate certificate databases for storing keys and trusted keys is used. The relationship of these two certificates databases or keyfiles is shown in Figure 11-2. Here, the server's KeyFile.jks contains a private/public certificate key pair, signed by a third party Certificate Authority (CA). The TrustFile.jks

associated with the server can then be used to contain only signer certificates from trusted peers. The reverse is true for any client, hence the client's KeyFile.jks and client's TrustFile.jks files. It also remains possible to use a single file for both tasks.

The JSSE SSL authentication mechanism in WebSphere V4.0, however, has the requirement shown in Figure 11-3 for replacing the Demo Keyring. Note that the public certificate of the signing Certificate Authority (CA) is present in the client's KeyFile.jks keyfile and not the client's TrustFile.jks keyfile.

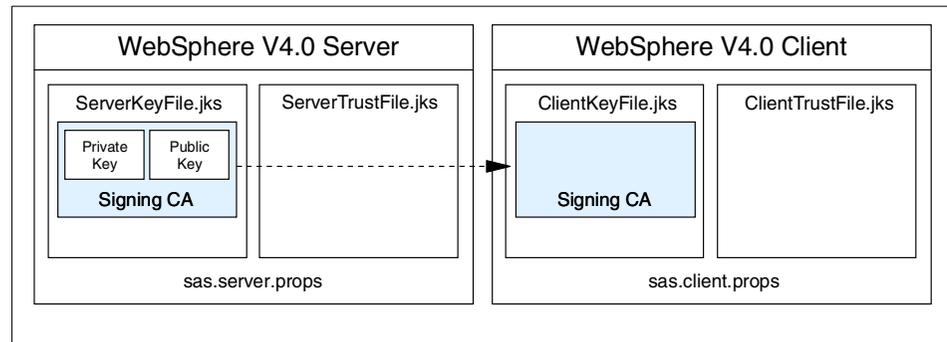


Figure 11-3 WebSphere keystores when using a third-party CA

Furthermore, if you opt to create a self-signed certificate and sign it yourself, the public key must be installed back into the server's KeyFile.jks keyfile as a trusted signer and not only into the server's TrustFile.jks keyfile. This is shown in Figure 11-4.

All other trusted client certificates can and should be installed into the WebSphere server's TrustFile.jks keyfile, as depicted in Figure 11-2 on page 217, if you elect to use the Global Default SSL configuration settings. One example of a client certificate is the public certificate used by the IBM SecureWay LDAP Directory Server, that allows WebSphere to communicate over LDAPS. In this case, the public certificate is required in the server's TrustFile.jks keyfile (only if you opt to use the default SSL settings when configuring LDAPS).

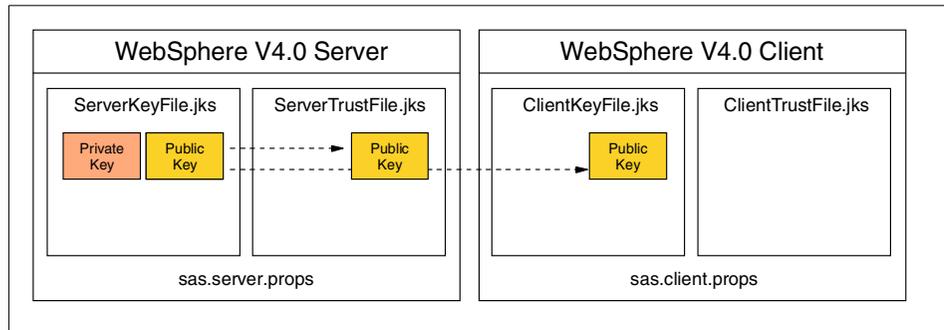


Figure 11-4 WebSphere keystores when using a self-signed certificate

The client's KeyFile.jks and client's TrustFile.jks keyfiles are used by the WebSphere Administration Client and other J2EE clients. Such clients reference the sas.client.props file in the WebSphere properties directory for authentication.

The following options are discussed here:

1. Self-signed certificate using the IBM ikeyman utility
2. Certificate signed by a third-party CA

11.1.2 Option 1: self-signed certificate using the IBM ikeyman utility

The IBM ikeyman tool offers the WebSphere administrator the ability to create and manage digital certificates. WebSphere ships with an updated JSSE ikeyman that supports the Java Key Store format (JKS). In the example that follows, we will use ikeyman to create a private/public key pair self-signed certificate to replace the Demo Keyring present in the DummyServerKeyFile.jks keyfile.

The server's KeyFile.jks certificate database

Create a new server keyfile using the WebSphere V4.0 JSSE ikeyman tool:

1. Launch the JSSE ikeyman tool by invoking it directly from the WebSphere bin directory; on Unix platforms, invoke **ikeyman.sh**. On Windows, the tool can be launched from the Start menu.
2. From the ikeyman menu bar, select **Key Database file -> New**.

Ensure that the Key database type is set to JKS.

- The file name can be any arbitrary name, but for clarity it is recommended that you standardize your naming convention. In Figure 11-5, the name `WASV4ServerKeyFile.jks` is used. Later on, we will be creating three more associated keyfiles:

`WASV4ServerTrustStore.jks`
`WASV4ClientKeyFile.jks`
`WASV4ClientTrustStore.jks`

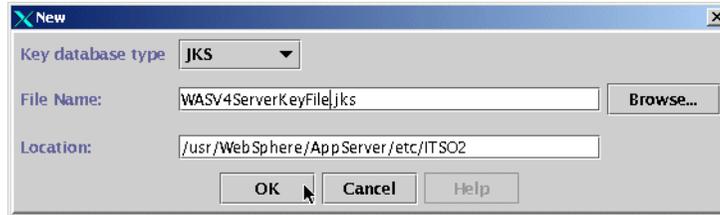


Figure 11-5 New `WASV4ServerKeyFile.jks`

- Click **OK** when you are finished.
- Enter a password when prompted.

The password strength is governed by the randomness of the characters you select. Click **OK**.

- Although this is optional, we suggest that you delete all of the Root Certificate Authority (CA) certificates found under the Signer Certificates menu in ikeyman.

As we will be creating a self-signed certificate to secure a peer-to-peer connection, there is no requirement to authenticate against the public key of a trusted third-party Certificate Authority (CA).

- If you have decided to delete the certificates, then select the certificates one by one and click **Delete**. Repeat this step until all Root Certificate Authorities have been deleted.

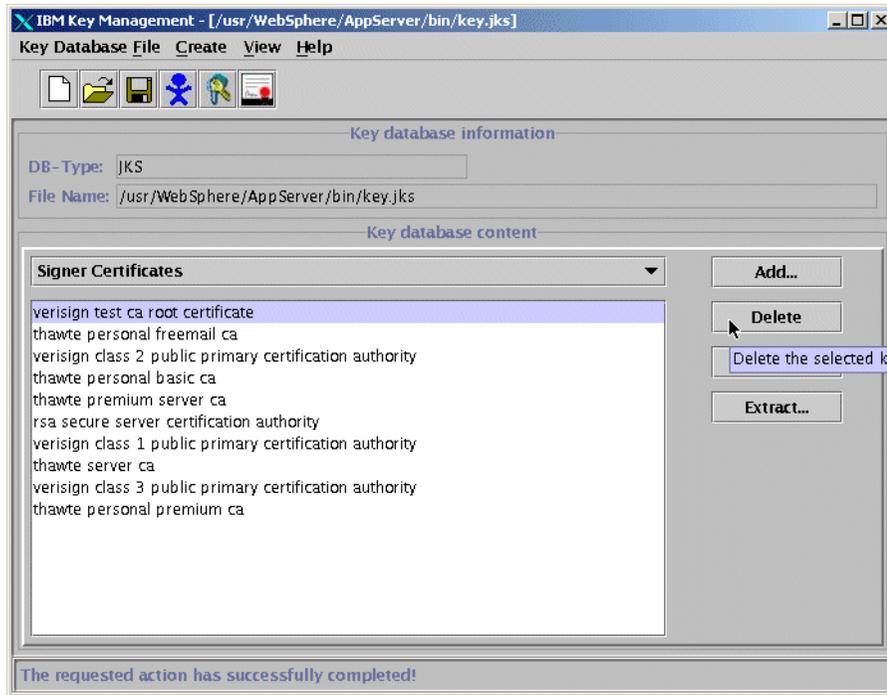


Figure 11-6 IBM Key Management window

8. Next, create a new self-signed certificate key pair. From the JSSE ikeyman tool, select **Create > New Self-Signed Certificate**.
9. Set the Key Label name to WASV4IntSec or a similar identity. We suggest that you not use any spaces in the name.
10. The certificate Version and Key Size can remain unmodified at X509 V3 and 1024, respectively.
11. The first two fields of the Subject Name or Distinguished Name are mandatory and must be completed. Likewise, the Country code and Validity Period must be set. For completeness, you should, however, fill in all the fields.
12. Click **OK** when you are finished.

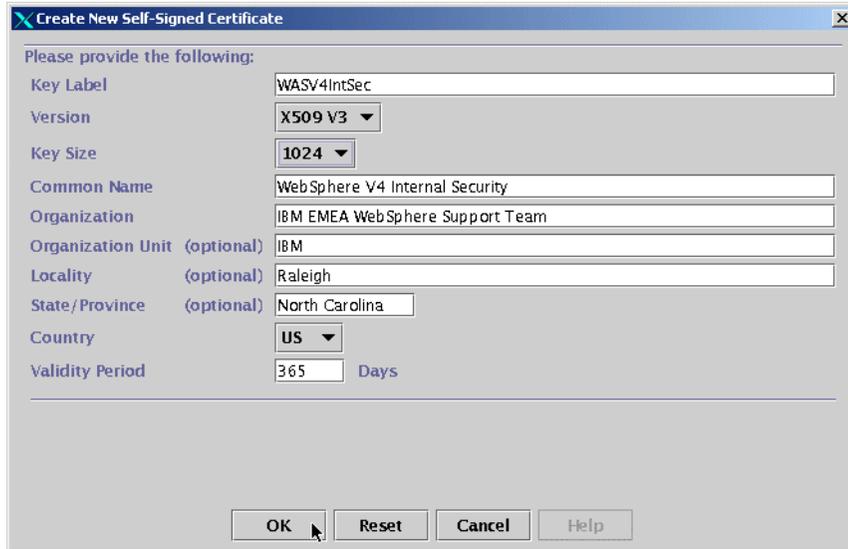


Figure 11-7 Create New Self-Signed Certificate window

13. Extract the newly created public key.

As we will be wanting to perform authentication against the newly created private/public self-signed certificate, we must first extract the public key from the WASV4ServerKeyFile.jks keystore.

14. Select the previously created certificate under the ikeyman **Personal Certificates** drop-down menu and click the **Extract Certificate** button.
15. Ensure that the Data type is set to Base64-encode ASCII data for adherence to the Internet RFC 1421 standard and set the extracted Certificate file name to WASV4IntSecPubCert.arm or a similar value; this is the location where the file was saved, in our case, /usr/WebSphere/AppServer/etc. Click **OK**.
16. Finally, add the extracted public key as a trusted signer back into the keyfile.
For self-authentication, the previously extracted public key must be imported back into the WASV4ServerKeyFile.jks keyfile as a trusted signer. This is equivalent to being your own Certificate Authority (CA).
17. Select **Signer Certificates** from the JSSE ikeyman drop-down menu and click the **Add** button.
18. Ensure that the Data type is set to Base64-encode ASCII data for compatibility and select the file containing the public key previously extracted, **WASV4IntSecPubCert.arm** or the file name of your choice, to the location: /usr/WebSphere/AppServer/etc.
19. Click **OK** when you are finished.

20. You will then be prompted to enter a Label Name for the newly installed trusted signer certificate, in our example: `internalsecurity`. This can be any arbitrary name, without any spaces, except that of an already existing label name. It cannot be the same name as the name of the personal certificate originally created.
21. Click **OK** when you are finished.

Important: We found that under certain circumstance, SSL handshaking will fail if the label name for the certificate contains any spaces.

At this point, it is a good idea to create the three additional certificate databases/keyfiles that are needed in the implementation.

The server's TrustFile.jks certificate database

You may choose to create a Server trust file at this point. However, there is no requirement to place any certificates into the certificate database/keyfile.

1. Create a new keyfile using the WebSphere V4 ikeyman tool.
Repeat steps 1 to 7 from “The server's KeyFile.jks certificate database” on page 219 as used for creating the WASV4ServerKeyFile.jks keyfile, but instead name the keyfile WASV4ServerTrustFile.jks.
2. When this is completed, close the empty certificate database.

The client's KeyFile.jks certificate database

To enable the WebSphere Administration Client or any other J2EE Client to securely communicate with the WebSphere Administration Server, install the previously extracted public self-signed certificate WASV4IntSecPubCert.arm as a trusted signer into the Client certificate database/keyfile.

1. Create a new Client Keyfile using the WebSphere 4 ikeyman tool.
Repeat steps 1 through 7 from “The server's KeyFile.jks certificate database” on page 219 as used for creating the WASV4ServerKeyFile.jks, but instead name the keyfile WASV4ClientKeyFile.jks.
2. Then, in the Signer Certificates ikeyman drop-down menu, click the **Add** button.
3. Ensure the Data type is set to Base64-encode ASCII data for compatibility and select the file containing the public key previously extracted, WASV4IntSecPubCert.arm or the file name of your choice.

4. You will then be prompted to enter a Label Name for the newly installed trusted signer certificate. This can be any arbitrary name, without any spaces, except that of an already existing label name.
5. When this is completed, close the certificate database/keyfile.

The client's TrustFile.jks certificate database

For completeness, you can choose to create a certificate database to represent the Client trust file, similar to what was created for the Server trust file: simply follow the steps documented for creating the server's TrustFile.jks above, but instead name the certificate database WASV4ServerTrustFile.jks.

You should now have the following four certificate databases/keyfiles:

```
WASV4ServerKeyFile.jks
WASV4ServerTrustFile.jks
WASV4ClientKeyFile.jks
WASV4ClientTrustFile.jks
```

11.1.3 Option 2: certificate signed by a third-party CA

As discussed previously, it is quite acceptable to use a certificate key pair signed by a third-party Certificate Authority (CA) to replace the Demo Keyring present in the DummyServerKeyFile.jks file.

In the example that follows, we demonstrate how it is possible to generate an SSL certificate suitable for signing by a CA, using the Java keytool command line utility. If you prefer to use the IBM ikeyman tool for this step, ensure that you use the Java Key Store (JKS) compatible version that ships with WebSphere and follow the instructions for configuring a Web server certificate detailed in Chapter 21 of the *WebSphere V4.0 Handbook*, SG24-6176.

Note: The steps are identical for creating the certificate, even though you are not securing a Web server or using the IBM ikeyman tool that supports the CMS keystore format.

It is suggested that you name the keyfile WASV4ServerKeyFile.jks, to maintain consistency with the example detailed in this section. Additionally, there is no requirement to save the keyfile password to a stash file when using the JKS compatible ikeyman tool.

Before proceeding with the Java command line keytool, set the environment to ensure that the keytool executable is in the PATH. Then change to the directory where you wish to generate the keyfile and associated files.

As such, on AIX:

```
#export PATH=$PATH:/usr/WebSphere/AppServer/java/jre/bin/  
#cd /usr/WebSphere/AppServer/properties/com/ibm/websphere/etc/itso
```

Generating your server's KeyFile.jks certificate database

The following steps will guide you through the process of generating the server's KeyFile.jks certificate database.

1. Generate a self-signed Key.

The command shown in Example 11-1 generates a private/public self-signed certificate key pair, where the owner (or entity) and issuer (signer) are the same. As the signer algorithm is based on the key algorithm, specifying RSA for the private key ensures that MD5withRSA will be used, rather than SHA1withDSA (the default). This was found to be a necessary option to successfully import any public certificate key generated with keytool into the IBM GSKIT ikeyman tool that ships with the IBM HTTP Server (IHS) and the IBM SecureWay LDAP Directory Server.

Example 11-1 Keytool key generation

```
#keytool -genkey -keyalg RSA -dname "cn=WebSphere V4 Internal Security, OU=IBM  
EMEA WebSphere Support Team, o=IBM, c=GB" -alias WSV4IntSec -keypass websphere  
-keystore WSV4ServerKeyFile.jks -storepass websphere -validity 365
```

The Subject Name or Distinguished Name in the above example is set as follows:

```
cn=WebSphere V4 Internal Security, OU=IBM EMEA WebSphere Support Team,  
o=IBM, c=GB
```

Care should be taken when specifying the Distinguished Name, as the Common Name (CN) field is often misinterpreted. For example, VeriSign Server IDs are specific to the CN and must be based on a fully qualified host name. VeriSign also stipulates that the State/Province field be completed, in which case you should ensure that the CN takes the following format, when using VeriSign as your chosen CA (this was not found to be an issue when requesting a test certificate from Thawte Consulting):

```
cn=rs617001.itso.ra1.ibm.com, OU=IBM EMEA WebSphere Support Team, o=IBM,  
c=GB
```

Finally, an alias of WSV4IntSec is associated with the newly created private/public certificate key pair. You also need to set the password associated with the private key (keypass) and the password protecting the actual keystore (storepass). The validity is set to 365 days and, because no keysize is specifically set, the default of 1024 is used.

2. Generate a Certificate Signing Request for submission to a third party CA.

The next step is to produce a Certificate Signing Request (CSR) based on the previously created self-signed certificate. This will create a CSR ready for submission to a CA, in the BASE64-encoded ASCII data format.

Example 11-2 keytool CSR generation

```
#keytool -v -certreq -alias WSV4IntSec -file WSV4IntSecReq.csr -keypass  
websphere -keystore WSV4ServerKeyFile.jks -storepass websphere
```

In the above example, the Certificate Signing Request (CSR) is saved to a file named `WSV4IntSecReq.csr`, in the current working directory. It is this file which, when viewed, can be cut and pasted into the CA online signing tool. As such, your resulting CSR will look similar to that as shown in Example 11-3, using the following command: **more WSV4IntSecReq.csr**

Example 11-3 WSV4IntSecReq.csr example

```
-----BEGIN NEW CERTIFICATE REQUEST-----  
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCROIXDDAKBgNVBAoTA01CTTEoMCMYGA1UECmFmSUJNIEVN  
RUEgV2ViU3BoZXJlIFN1cHBvcnQvGVhbTEuMCUGA1UEAxMev2ViU3BoZXJlIFY0IE1udGVybWFs  
IFN1Y3VyaXR5MIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCI4m1BLZ8xn5PQHdu+LEOFroyL  
JXOMsUvOUvKcOie6q/z4GqnJzCW0ejbDTAbrPUTrWxkSxo4T5vMkIOdsEEbvNvNYAGIh1J6wE1r7  
DaQ/DhxLadDUTHH2+zc1cSTFCoC0eb7g7N14/RVrXB7o/M5pyptzCrMn7BUkkUgn/UFB1QIDAQAB  
oAAwDQYJKoZIhvcNAQEEBQADgYEAwTUrQdUO/wHEvzGFELATpVR4PgYM9URrjiZ4d51seMFOONb6  
1nUvQxxNdZkB9KNM1qrMbKYvRC90vTM1HgCi dunKJS/9aGv4SQId1hWLwZ0pLhiU8UH7YKsx8xU4  
g70TWoKGV7rbBfSmYghoB3ed0CRVg2mKvWbi du9Tp0322As=  
-----END NEW CERTIFICATE REQUEST-----
```

3. Import the Certificate authenticating the public key of the trusted CA.

The Java keytool, by default, does not place any trusted CA public certificate keys into your keyfile. For this reason, you need to download the CA's public root certificate key, corresponding to that used by the CA to sign your CSR. It is strongly recommended that you cross-check the credentials of this key before importing it into your keystore. The java keytool also ships with an associated *cacerts* file found under the `{WAS_ROOT}/java/jre/lib/security/` directory, as an alternative source for locating the most popular public root signing CA certificates in circulation.

In Example 11-4, Thawte's Test Root CA public certificate is imported under the alias of Thawte Test CA Root into our keyfile `WSV4ServerKeyFile.jks`.

When prompted to Trust the certificate, type *yes* only if you agree with the credentials displayed for the CA's public root certificate.

Example 11-4 Importing the CA root certificate

```
#keytool -import -alias "Thawte Test CA Root" -file ThawteTestCA.arm -keystore  
WSV4ServerKeyFile.jks -storepass websphere
```

```
Owner: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Issuer: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Serial number: 0
Valid from: Wed Jul 31 20:00:00 EDT 1996 until: Thu Dec 31 16:59:59 EST 2020
Certificate fingerprints:
    MD5: 5E:EO:0E:1D:17:B7:CA:A5:7D:36:D6:02:DF:4D:26:A4
    SHA1: 39:C6:9D:27:AF:DC:EB:47:D6:33:36:6A:B2:05:F1:47:A9:B4:DA:EA
Trust this certificate? [no]: yes
Certificate was added to keystore
```

4. Import the certificate reply from the Certificate Authority (CA).

Once you have received the reply from your chosen CA, you can import the CA signed response and, in doing so, replace your self-signed certificate key pair.

Typically, the response will take the form of an .arm file and will be in the BASE64-encoded ASCII data format.

Import the certificate response from the CA into the keyfile using the same alias name as first given to the self-signed certificate. In the example shown, this is WSV4IntSec. Using an alternative alias name will generate a new signer certificate and not a personal certificate chain as required.

Example 11-5 Importing the certificate response from the CA

```
#keytool -import -trustcacerts -alias WSV4IntSec -file WSV4IntSecRes.arm
-keystore WSV4ServerKeyFile.jks -storepass websphere
Certificate reply was installed in keystore
```

This concludes the steps necessary to create and sign the server certificate that will be used to replace the Demo Keyring.

You can check the contents of your certificate database/keyfile at any time by using the *keytool* list option to view all of your certificate entries, as shown in Example 11-6. You will find that the self-signed certificate wsv4intsec is now replaced by a chain of certificates, with the CA now authenticating the public key. There is also a separate entry listed for the trusted certificate belonging to the CA's public root certificate.

Example 11-6 Listing the contents of the keystore

```
#keytool -list -v -keystore WSV4ServerKeyFile.jks -storepass websphere
Keystore type: jks
Keystore provider: SUN
```

Your keystore contains 2 entries:

Alias name: wsv4intsec

```
Creation date: Mon Oct 15 17:08:54 EDT 2001
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=WebSphere V4 Internal Security, OU=IBM EMEA WebSphere Support Team,
O=IBM, C=GB
Issuer: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Serial number: 65f724
Valid from: Mon Oct 15 18:06:29 EDT 2001 until: Mon Nov 05 17:06:29 EST 2001
Certificate fingerprints:
    MD5: 75:61:23:E2:B4:8E:76:12:B9:B4:20:14:91:D8:39:97
    SHA1: 8F:45:46:13:44:14:B4:F6:F4:B7:D4:B7:8D:00:B6:65:CC:4E:62:2E
Certificate[2]:
Owner: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Issuer: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Serial number: 0
Valid from: Wed Jul 31 20:00:00 EDT 1996 until: Thu Dec 31 16:59:59 EST 2020
Certificate fingerprints:
    MD5: 5E:E0:0E:1D:17:B7:CA:A5:7D:36:D6:02:DF:4D:26:A4
    SHA1: 39:C6:9D:27:AF:DC:EB:47:D6:33:36:6A:B2:05:F1:47:A9:B4:DA:EA
*****
*****
Alias name: thawte test ca root
Creation date: Mon Oct 15 17:08:25 EDT 2001
Entry type: trustedCertEntry
Owner: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Issuer: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Serial number: 0
Valid from: Wed Jul 31 20:00:00 EDT 1996 until: Thu Dec 31 16:59:59 EST 2020
Certificate fingerprints:
    MD5: 5E:E0:0E:1D:17:B7:CA:A5:7D:36:D6:02:DF:4D:26:A4
    SHA1: 39:C6:9D:27:AF:DC:EB:47:D6:33:36:6A:B2:05:F1:47:A9:B4:DA:EA
*****
*****
```

Generating your client's KeyFile.jks certificate database

The only requirement that now exists is to import the same public root certificate key associated with the Certificate Authority (CA) authenticating what will be the WebSphere server's public key into the client certificate database/keyfile. This accomplishes the requirement shown in Figure 11-3 on page 218.

1. Import the root certificate into a new certificate database. The CA's public root certificate authenticates the previously generated server public certificate key.

The action above creates a new keyfile called WSV4ClientKeyFile.jks and sets the password protecting the keyfile (storepass) to websphere.

2. When prompted to Trust the certificate, type **yes** only if you agree with the fingerprints displayed for the CA's public root certificate.

Example 11-7 Creating the associated WSV4ClientKeyFile.jks

```
#keytool -import -alias "Thawte Test CA Root" -file ThawteTestCA.arm -keystore
WSV4ClientKeyFile.jks -storepass websphere
Owner: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Issuer: CN=Thawte Test CA Root, OU=TEST TEST TEST, O=Thawte Certification,
ST=FOR TESTING PURPOSES ONLY, C=ZA
Serial number: 0
Valid from: Wed Jul 31 20:00:00 EDT 1996 until: Thu Dec 31 16:59:59 EST 2020
Certificate fingerprints:
    MD5:  5E:E0:0E:1D:17:B7:CA:A5:7D:36:D6:02:DF:4D:26:A4
    SHA1: 39:C6:9D:27:AF:DC:EB:47:D6:33:36:6A:B2:05:F1:47:A9:B4:DA:EA
Trust this certificate? [no]: yes
Certificate was added to keystore
```

This concludes the steps necessary to create certificate database/keyfile that will be used by the WebSphere Administrative Client and other J2EE clients when communicating securely with the WebSphere Administrative Server.

11.1.4 Configuring WebSphere to use your own keyring

This section assumes you have either created a private/public self-signed certificate key pair with the public key set as a trusted signer, or a private/public certificate key pair signed by a third-party Certificate Authority (CA). At this point, it is recommended that Global Security not be enabled.

Important: Before modifying any of the default security settings, copy the `sas.client.props` and `sas.server.props` files found under the WebSphere properties directory to a safe place. You may also choose to take an XMLConfig export of your WebSphere administration repository, if you have already deployed applications or made significant modifications.

To enable WebSphere to use your own keyring, follow these steps:

1. Start WebSphere without Global Security enabled and launch the Administrative Console. If Global Security is enabled, temporarily disable it while you perform the following steps. This will involve stopping and starting the WebSphere Application Server to disable Global Security.
2. From the WebSphere Administrative Client menu bar select **Console > Security Center...**
3. Once at the Security Center General window, select the **Default SSL Configuration** button to view the SSL parameters used for Global Security.

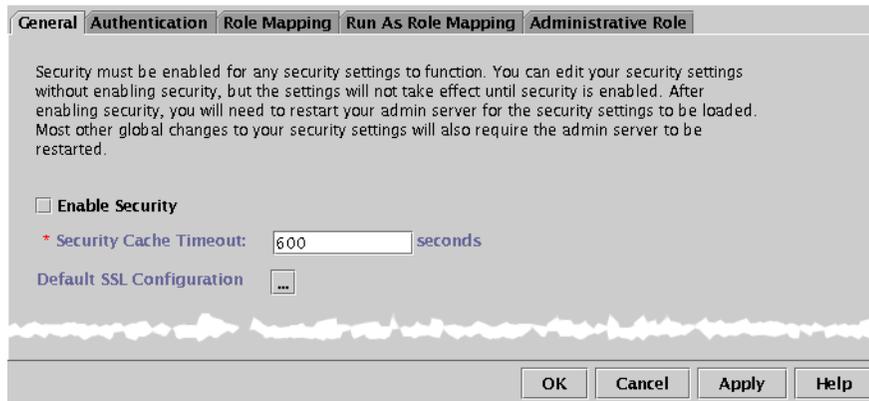


Figure 11-8 WebSphere Global Security General window

4. Figure 11-9 shows the default parameters specified under the Default SSL Configuration settings. Here, the SSL key and trust file fields are set to use the DummyServerKeyFile.jks and the DummyServerTrustFile.jks certificate databases/keyfiles, respectively. For both files, the password is set to WebAS, although this is not shown in plain text.

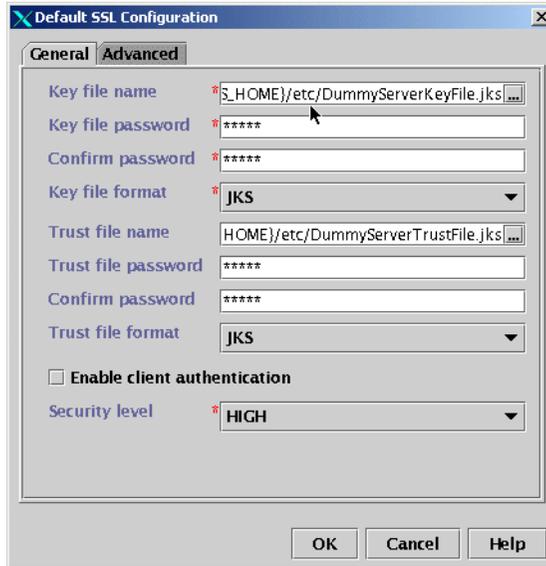


Figure 11-9 Default SSL configuration

5. Proceed by modifying the default SSL parameters to reference your newly created Java Key Store (JKS) certificate databases/keyfiles. You may choose to use a single JKS certificate database for both your personal keys and trusted signer certificate keys. In that case, set both the Key file and Trust file fields to the same value. We modified the fields to use the values shown below:

Key file name: (WAS_HOME)/etc/WASV4ServerKeyFile.jks

Key file password: webspHERE

Confirm password: webspHERE

Key file format: JKS

Trust file name: (WAS_HOME)/etc/WASV4ServerTrustFile.jks

Key file password: webspHERE

Confirm password: webspHERE

Key file format: JKS

Security level: High

Note: The password will not appear in the text field for security reasons, showing a * (an asterisk) instead of the characters.

6. Click **OK** when you are done.

Depending on how you modify the Key file and Trust file fields, you may be presented with the SECJ8313W warning message. If you are not concerned with multiple domains, that is, if your WebSphere install is a single instance, typing in the fully qualified path to the WebSphere root directory in place of (WAS_ROOT) and then clicking the **Tab** button will allow you to search the relevant directories.

Note: SECJ8313W: using the symbolic link \$(WAS_ROOT) is recommended, especially in a multiple admin servers configuration. Preserving the symbolic link is not always possible when file browser is used to select a file. Hence using of browser is disabled when symbolic link is present.

Alternatively, retain the (WAS_ROOT) symbolic link and specify the remaining path to your relevant Java Key Store (JKS) certificate databases. In this case, the certificate databases must reside in a directory below (WAS_ROOT), the WebSphere root directory. On AIX this is /usr/WebSphere/AppServer.

Another option is to use XMLConfig to set the default SSL configuration parameters. An example of the necessary XML is shown in Example 11-8, with the respective XMLConfig command being: `./XMLConfig.sh -adminNodeName rs617001 -import itsosec.xml`

Example 11-8 itsosec.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///XMLConfigDTDLocation$$dsep$xml config.dtd" >
<websphere-sa-config>
<security-config security-cache-timeout="600" security-enabled="false">
<ssl-config>
  <key-file-name>${WAS_HOME}/etc/ITSO/WSV4ServerKeyStore.jks</key-file-name>
  <key-file-password>websphere</key-file-password>
  <key-file-format>0</key-file-format>
  <client-authentication>false</client-authentication>
  <security-level>0</security-level>
  <crypto-hardware-enabled>false</crypto-hardware-enabled>
  <crypto-library-file></crypto-library-file>
  <crypto-password>{xor}</crypto-password>
  <crypto-token-type></crypto-token-type>
</trust-file-name></trust-file-name>
```

```
<trust-file-password></trust-file-password>
<property name="com.ibm.ssl.protocol" value="SSLv3"/>
</ssl-config>
</security-config>
</websphere-sa-config>
```

The password will automatically be converted from plain text to a hashed algorithm when imported. This example demonstrates that it is possible to retain the `#{WAS_HOME}` modifier in the key-file-name file.

7. Click **OK** to return to the main General Security window.
8. Click **Apply** to instigate the modifications and then **OK** to close the General Security window. Ensure that WebSphere Global Security remains disabled at this point.
9. Restart WebSphere to ensure that the modifications take effect. If you examine the WebSphere trace file, you will see references to keyStore and trustStore being updated:

```
SECJ0046E: SAS Property:com.ibm.ssl.keyStore has been updated
SECJ0046E: SAS Property:com.ibm.ssl.trustStore has been updated
```

Similarly, the `sas.server.props` and `sas.server.props.future` files will be updated to reference the new Java Key Store (JKS) certificate databases. If the `sas.server.props` file fails to show the expected modifications, you will need to go through the previous steps again, ensuring that you click **Apply** in the WebSphere Security Center General window. The following example shows an excerpt from the file `sas.server.props`.

Example 11-9 sas.server.props excerpt

```
....
com.ibm.ssl.trustStore=/usr/WebSphere/AppServer/etc/WASV4ServerTrustFile.jks
com.ibm.ssl.clientAuthentication=false
com.ibm.ssl.keyStore=/usr/WebSphere/AppServer/etc/WASV4ServerKeyFile.jks
com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.keyManager=IbmX509
com.ibm.ssl.trustStoreType=JKS
com.ibm.ssl.keyStorePassword={xor}CDo9Hgw\=
com.ibm.ssl.trustManager=IbmX509
com.ibm.ssl.tokenType=PKCS\#11
com.ibm.ssl.protocol=SSLv3
com.ibm.ssl.trustStorePassword={xor}CDo9Hgw\=
....
```

10. You are now set to enable WebSphere Global Security as detailed in Section 11.1.6, “Enabling Global Security and securing the Administrative Server” on page 235, and to test your Keyring.

Tip: If you are testing your own Keyring for the first time, we advise that you use LocalOS as the selected authentication mechanism when enabling WebSphere Global Security. If you can, however, guarantee that an alternative authentication mechanism is known to work, then you may test your Keyring against that registry. This way, problems can be isolated to the Keyring (certificate database).

11.1.5 Modifying the sas.client.props file

Supplementing your own Keyring in place of the Demo Keyring will prevent the WebSphere Administration Client and any other configured J2EE Client from successfully connecting to the WebSphere Administrative Server after Global Security is enabled.

ADGU2008E: the Administration Client failed to connect to the Administration Server. Start the local or remote Administration Server service before launching the Administration Client.

Before you can successfully launch the WebSphere Administrative Client, you must modify the sas.client.props file to reference the corresponding Java Key Store (JKS) certificate database that you created earlier. It will either contain the public self-signed certificate key associated with the WebSphere Administration Server, or if a third-party Certificate Authority (CA) was used, the trusted root certificate of the CA authenticating the WebSphere Application Server public key.

In the following steps we will be using the WASV4ClientKeyFile.jks and WASV4ClientTrustFile.jks Java Key Store (JKS) certificate databases/keyfiles previously created in Section 11.1.2, “Option 1: self-signed certificate using the IBM ikeyman utility” on page 219.

1. Using your favorite editor, modify the sas.client.props file in the (WAS_ROOT)/properties directory to reference the previously created certificate databases. Ensure that the password used to protect each database is specified appropriately. As such, the edited sas.client.props file will look similar to that shown below in Example 11-10.

Example 11-10 Modified sas.client.props file

```
...
com.ibm.ssl.protocol=SSL
com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.keyStore=/usr/WebSphere/AppServer/etc/WASV4ClientKeyFile.jks
com.ibm.ssl.keyStorePassword=WebAS
com.ibm.ssl.trustStoreType=JKS
com.ibm.ssl.trustStore=/usr/WebSphere/AppServer/etc/WASV4ClientTrustFile.jks
```

```
com.ibm.ssl.trustStorePassword=WebAS
```

```
...
```

2. Note that if you have not yet enabled WebSphere Global Security, the following directive in the `sas.client.props` will be set to `false`.

```
com.ibm.CORBA.securityEnabled=true
```

However, if Global Security is enabled at the WebSphere Administrative Server, the directive should read `true`, as shown. Usually, this adjustment is made automatically by WebSphere when Global Security is enabled.

If for some reason the adjustment is not made when Global Security is enabled, you will either see the ADGU2008E message on the last page or the security error shown below. If either error occurs, check the `securityEnabled` directive in the `sas.client.props` file and modify accordingly.

ADGU2009E security error: either the user name/password is wrong or this user is not authorized to connect to admin server.

3. After modifying the `sas.client.props` file, save your changes.

11.1.6 Enabling Global Security and securing the Administrative Server

This section provides instructions for enabling WebSphere Global Security, an action which, by default, secures the WebSphere Administrative Server and any managed Application Servers configured on the same node. Administrators should recall that there are additional steps for securing individual applications and resources served by WebSphere; these tasks are addressed in other chapters of this redbook.

Enabling WebSphere Global Security

1. Start the WebSphere Administrative Client and select the **Security Center** from the Console drop-down menu.
2. Under the Security Center **General** tab, click the **Enable Security** button.
3. The Security Cache Timeout setting can be adjusted to extend or reduce the period of time that security credential information is retained by WebSphere. Once the cache expires, WebSphere will have to re-query the selected user registry.
4. Clicking the **Default SSL Configuration** button will allow you to modify the Java Key Store (JKS) certificate databases, used by the underlying Secure Socket Layer (SSL) transport mechanism. We covered this task previously

under Section 11.1.4, “Configuring WebSphere to use your own keyring” on page 229.



Figure 11-10 WebSphere V4.0 Security Center General window

5. Once you have enabled WebSphere Global Security, select the **Authentication** tab from the Security Center and select your desired authentication mechanism. As such, you will need to select your desired authentication mechanism from one of the following three options:
 - Local Operating System Authentication
 - LTPA Authentication
 - Custom User Registry mechanism

Option 1: Local Operating System Authentication mechanism

You can select to authenticate WebSphere users against the local operating system user registry or the /etc/passwd password file. Note that this option is not supported if you elect to run WebSphere as a non-root user. In that case, you can select to use the LTPA authentication mechanism or Custom Registry option.

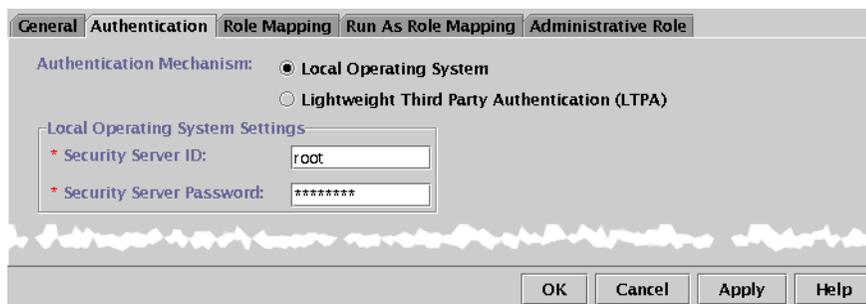


Figure 11-11 Local Operating System authentication

1. Complete the Security Server ID and associated password fields accordingly; this is the identity with which WebSphere will run.
2. Click **Apply** and then **OK**.
3. Restart WebSphere to ensure that the modifications take effect.

Option 2: the LTPA Authentication mechanism

If you opt to use the Lightweight Third Party Authentication (LTPA) mechanism, you must choose between either the LDAP or Custom User Registry. If you choose to authenticate against a remote LDAP Directory Server, complete the following steps:

The screenshot shows the 'Authentication' tab of the WebSphere Security Configuration dialog. The 'Authentication Mechanism' is set to 'Lightweight Third Party Authentication (LTPA)'. Under 'LTPA Settings', 'Token Expiration' is 120 minutes, 'Enable Single Sign On (SSO)' is checked, and the 'Domain' is 'itso.ral.ibm.com'. Below this, there are buttons for 'Generate Keys...', 'Import Key...', and 'Export Key...'. The 'LDAP' radio button is selected, and the 'LDAP Settings' section is expanded. It shows 'Security Server ID' as 'websphere', 'Port' as '636', 'Security Server Password' as '*****', 'Base Distinguished Name' as 'ou=uk,dc=interne', 'Host' as '1.itso.ral.ibm.com', 'Bind Distinguished Name' as 'cn=websphere_ou', and 'Bind Password' as '*****'. The 'Directory Type' is set to 'SecureWay'. There are buttons for 'Advanced...' and 'SSL Configuration' at the bottom of the LDAP settings. At the very bottom of the dialog are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

Figure 11-12 LTPA settings

Complete the (optional) LTPA settings as follows:

1. **Token Expiration:** specify the duration for which the LTPA token will be valid, set to 120. The LTPA token is encrypted using the LTPA Keys and gets passed to a client in the form of a HTTP cookie. Clients are forced to re-authenticate once the LTPA token expires.
2. **Enable Single Sign On (SSO):** use this if you wish to permit users to move between different applications/Web resources, located on the same or

different physical servers, without being prompted to log in each time a subsequent resource is requested.

3. **Domains:** specify the DNS domain name for the scope of the Single Sign On (SSO) operation. Note that all participating SSO servers must be in the same domain. IP addresses are not acceptable here. Our domain was: `itso.ral.ibm.com`.
4. **Limit to SSL connections only** will restrict Single Sign On (SSO) to only HTTPS requests originating from a client Web browser.
5. **Enable Web Trust Association:** use this if you wish to delegate the authentication of incoming requests to a third-party application. There must, however, be a mutual level of trust established between WebSphere and the third-party application, such as the Tivoli Policy Director product. Further details on this topic are explained in Chapter 13, “Policy Director” on page 353.
6. **Create Keys..., Import Keys..., Export Keys...:** to configure Single Sign On (SSO) across multiple Application Servers, all participating servers must share the same LTPA Keys. Note that key generation must be performed only after the LDAP settings, as discussed below, are confirmed, as the LDAP Directory Server host name is present in the exported file/LTPA key.

In addition to the LTPA settings, you must configure the LDAP settings specific to your LDAP Directory Server configuration. The settings and expected values are discussed in detail below (supplement your own values accordingly):

7. **Security Server ID:** specify the user you have created in your LDAP implementation that will represent the WebSphere Server identity, in our case: *websphere*.
8. **Security Server Password:** specify the corresponding password for the Security Server user; in our case, we used the password *websphere*.
9. **Host:** this is the fully qualified host name of the LDAP Directory Server, in our example: `rs617001.itso.ral.ibm.com`.
10. **Directory Type:** select **SecureWay** here. Note that if you modify any of the filtering rules, the Directory Type will change to *Custom* even though *SecureWay* remains the LDAP directory of your choice.
11. **Port:** specify 389, which corresponds to the TCP/IP port listening for LDAP queries on the remote *SecureWay* LDAP directory.

Note: in Figure 11-12, the port is set to 636 for secure LDAPS communication. Optionally, set this value to your own setting, if the remote Directory Server listens on a non-standard port.

12. **Base Distinguished Name:** set this to the entry point into your LDAP Directory Server where WebSphere will search for authenticated users. For our example we set: `ou=uk,dc=internetchaos,dc=com`.
13. **Bind Distinguished Name:** specify the fully qualified Distinguished Name of the LDAP user who has sufficient privileges to search and authenticate WebSphere users in the LDAP Directory Server. If you have restricted anonymous searches in the WebSphere user name space in your LDAP Directory, this field must be completed regardless of whether of not the user is the same as that specified for the Security Server ID. In this example, we used: `cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com`.
14. **Bind Distinguished Password:** specify the corresponding password for the Bind Distinguished user; our password was *websphere*.
15. If , upon clicking **OK**, the configuration fails with an error message or exception, double-check all parameters and certify that the remote LDAP Directory Server is up and running. Do not click **Apply** and do not terminate the Administration Client GUI at this point, as you will lose the ability to undo the changes that you made.
16. If your configuration proves successful, you can click **Apply** in the WebSphere Security Center window and then proceed to restart WebSphere for the changes to be included in the next runtime.

Option 3: the Custom User Registry authentication mechanism

The third authentication mechanism option available when configuring Global Security is the use of a Custom User Registry. As the implementation is somewhat configurable down to the specific code employed, it is discussed in further detail in 10.3, “CustomRegistry SPI” on page 206.

11.2 Configuring the Web Server to support HTTPS

With the demands of e-business, facilitating the secure flow of confidential material over the Internet is of paramount importance to every online organization. Not only is liability at stake, but also the reputation that can make or break a modern “dot com.” It comes as no surprise, then, that establishing a secure connection between a Web browser and Web server is probably the most widespread use for digital certificate technology in the world today.

In this section, we document the steps necessary to configure the IBM HTTP Server (IHS) to support HTTPS, the secure HTTP protocol.

Although we shall focus on the practical elements of configuring the IBM HTTP Server (IHS) and generating the appropriate Secure Socket Layer (SSL) Web server certificates, it is worth reiterating the role that a third-party Certificate Authority (CA) plays in the solution. Here, as shown in Figure 11-13 below, the Web browser has the CA public root certificate installed as a trusted entity. It is only because the Web server public key is signed and authenticated by the same trusted CA that the Web browser considers the Web server public key valid, in terms of identity, matching the claim substantiated by the CA.

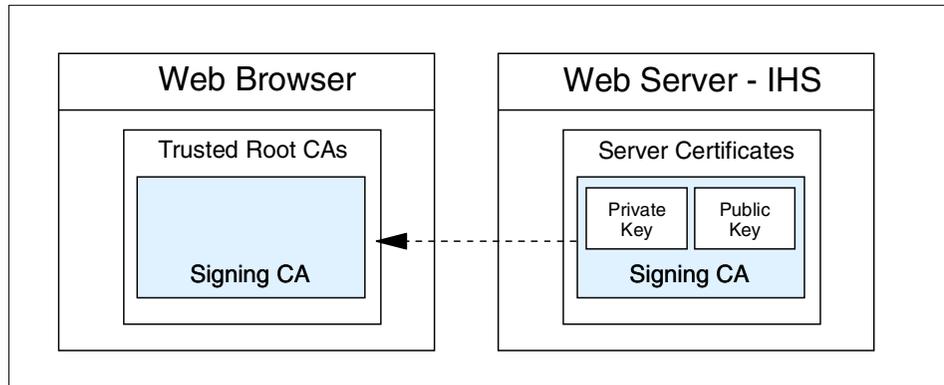


Figure 11-13 SSL certificate relationship for HTTPS

The diagram is not intended to show the flow of any certificate keys, but rather the location requirements for each certificate. For example, the Signing CA certificate must also be installed as a trusted entity in the client Web browser.

Note: Web browsers can still establish a secure connection if a Web server public key is not signed by a third-party CA. The risk is that the trust association offered by the CA is lost and potentially there is no way to corroborate that the public key associated with the Web server is genuine.

11.2.1 Generating a certificate to protect your Web Server

The instructions that follow highlight the tasks involved in creating an SSL server certificate for the IBM HTTP Server (IHS).

1. The starting point for creating a private/public certificate key pair to protect your Web server is the IBM ikeyman tool. Ensure that you invoke the ikeyman tool that ships as part of IBM HTTP Server (IHS) package, as it supports the CMS Key Database format (kdb).

This is not the IBM JSSE ikeyman version that ships under the WebSphere *bin* directory.

```
#export JAVA_HOME=/usr/jdk_base
```

```
#ikeyman
```

2. Create a new key database to store the certificate. You can manage multiple certificates in a single key database. From the ikeyman menu bar, select **Key Database File -> New**.
3. Set the following settings and click **OK** when you are done:
Key database file: CMS key database file
File name: IHS1319Certs.kdb
Location: /usr/HTTPServer/conf
4. At the Password Prompt window shown in Figure 11-14, enter the password of your choice. The more random the password, the higher the password strength. Finally, stash the password to a file so that the IHS can use the password to gain access to the certificates contained in the certificate key database.



Figure 11-14 Password prompt

5. From the IBM ikeyman menu bar, select **Create -> New Certificate Request** to create a new private/public certificate key pair.

Create New Key and Certificate Request

Please provide the following:

Key Label	IHS1319
Key Size	1024
Common Name	rs617001.itso.ral.ibm.com
Organization	EMEA WebSphere Support Team
Organization Unit (optional)	IBM
Locality (optional)	Research Triangle Park
State/Province (optional)	North Carolina
Zipcode (optional)	NC27709
Country	US

Enter the name of a file in which to store the certificate request:

/usr/HTTPServer/conf/IHS1319certreq.arm Browse...

Figure 11-15 Certificate details

Figure 11-15 above shows the options that we specified. You should supplement the fields accordingly to match your requirements. Click **OK** when this is completed.

Note: Consult the documentation from your chosen Certificate Authority (CA) prior to completing the fields. For example, VeriSign Server IDs stipulate that the Common Name (CN) must represent your fully qualified server name.

With the successful generation of your certificate and Certificate Signing Request (CSR), you will be prompted with the following message:

```
A new certificate request has been successfully create in the file....
You must send the file a certificate authority to request a certificate.
```

6. At this point, we strongly suggest that you close the certificate key database and quit ikeyman. In addition to the key database file and the previously generated Certificate Signing Request (CSR) file, you will find that you now have three extra files. The .sth suffixed file contains the password stash and will be used by the IHS to gain access to the certificate key database. The two remaining files (.crl and .rdb) contain internal information specific to the Certificate Signing Request (CSR). Back up all the files at this point as a precautionary measure.

```
#tar -cvf certificates.tar *
```

Important: If you accidentally destroy a Personal Certificate Request after you have submitted the Certificate Signing Request (CSR) to a Certificate Authority (CA), you will not be able to receive the response from the CA authenticating your public key into the certificate key database.

7. Submit the Certificate Signing Request (CSR) to a Certificate Authority (CA). Typically, this step is completed by cutting and pasting the CSR into the online tool provided by your selected CA. The data will look similar to the following:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB8jCCAVsCAQAwbExCzAJBgNVBAYTA1VTMRAwDgYDVQQREwd0QzI3NzA5MRcw
FQYDVQQIEw50b3J0aCBDYXJvYyB1uYTEfMBOGA1UEBxMUMVzZWYy2ggVHjYw5n
bGUgUGFyazEKMCIGA1UEChMbRU1FQSBXZlJcGh1cmUgU3VwcG9ydCBUZWFTMQww
CgYDVQQLEwNjQk0xIjAgBgNVBAMTGXJzNjE3MDAxLm10c28ucmFsLm1ibS5jb20w
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANnpW51HQhbLBb/FV70xYKgTG3wB
dYHrh5HKR8us1+fixRpm3AfwWUF4S4SsCz/fGZ6bT7fuA4k50ymHLz/ZN2Cg/8Z
G9K7qm2yvZIpZYrapLKhNArSVah85bFJKxAUYMW0z01aACaS4Rnkitow1n4NHG7E
Z/OvYUq77kfje159AgMBAAAgADANBgkqhkiG9w0BAQQFAA0BgQAICfcw73CnjDao
3p/AHi2ZxnOVxUznFpGwa1lQRrcsP2+B4VZ3mMeCiUZ8APVu5okxYE+C/k3nSY0g
92C+o0YesFGEgXiW4TZ/DJ56/zNWP1S18Wd1VQ9vwt5cKnA3LtNTJFTcDtmh0MNV
aoraXaT0voxAcS3Tgoe1E+5zEmWqHg==
-----END NEW CERTIFICATE REQUEST-----
```

8. In the period that it takes for the CA to respond in authenticating your public key, the certificate will be parked under the Personal Certificates Request menu in the IBM ikeyman utility.
9. Typically, you will receive the response from your chosen CA via e-mail. If the CA has adhered to the RFC 1421 standard, then the response will be in Base64-encoded ASCII data format. You can either save the attachment from your e-mail or cut and paste the response to a new file. In our example, we cut and pasted the data to a file called IHS1319certres.arm.
10. Invoke the IBM ikeyman tool again and open the key database previously created, entering the password when prompted.
11. Select the **Personal Certificates** menu and click the **Receive** button to take delivery of your now Certificate Authority (CA) signed public key. Set the following settings then click **OK**.
Data type: Base64-encoded ASCII data
Certificate file name: IHS1319certres.arm
Location: /usr/HTTPServer/conf/
12. If you selected to use a Certificate Authority (CA) that was not listed among the default group of Trusted Root Certification Authorities found under the Signer Certificates menu, you must use the **Add** option from the same menu, to import the public key associated with the authenticating CA, prior to

receiving any response to a Certificate Signing Request (CSR) into your key database.

Note: Any certificate that you introduce as a trusted entity into your key database poses a security risk. It is strongly recommended that you confirm the authenticity of such a certificate and cross-verify the certificate fingerprints with the identity you are trusting, using some alternative method.

13. You can verify the certificate now signed by the third-party Certificate Authority (CA) by double-clicking the associated label name as listed under the ikeyman Personal Certificates menu. This is shown in Figure 11-16.

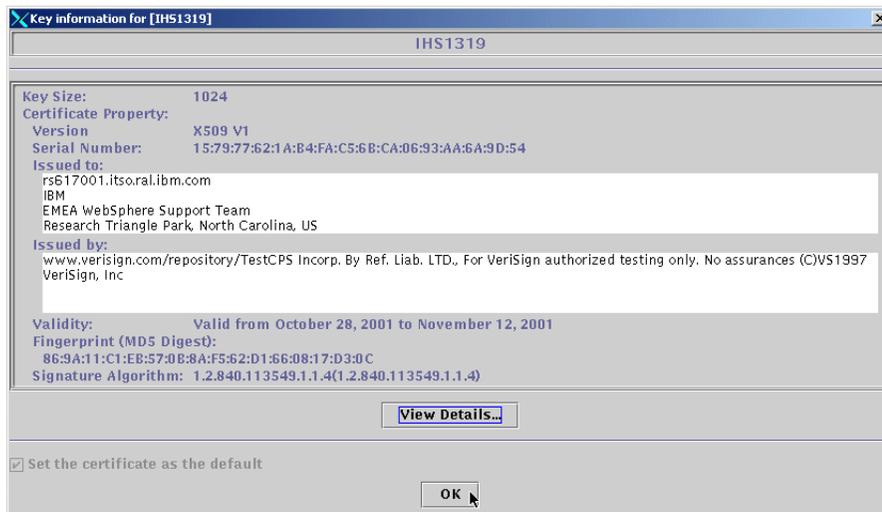


Figure 11-16 Verifying the certificate

This concludes the steps necessary to generate and sign the certificate that will be used to protect your IBM HTTP Web Server. You can now close the certificate database and terminate the ikeyman tool.

11.2.2 Configuring the IBM HTTP Server for SSL/HTTPS support

Although the IBM HTTP Server (IHS) is powered by Apache, one of the enhancements delivered in the IBM product is the level of SSL support. Here, IBM ships support for both SSL version 3, SSL version 2 and TLS version 1.

Prior to installing your Web server certificate, you must configure the IHS to load the IBM SSL modules at runtime. If you use the publicly distributed Apache product, you are required to install the OpenSSL module, as well as either ModSSL or Apache-SSL. Obviously, IBM only supports the IBM HTTP Server and the IBM SSL modules.

On Unix, you are required to add the following two directives to your `httpd.conf` file for SSL support:

```
LoadModule ibm_ssl_module      libexec/mod_ibm_ssl_128.so
AddModule mod_ibm_ssl.c
```

On Windows, only a single library needs to be loaded in the `httpd.conf` file:

```
LoadModule ibm_ssl_module modules/IBMModuleSSL128.dll
```

Note that on AIX the IBM SSL module supporting 128-bit encryption is delivered in the `http_server.ssl.128.1.3.19.0` fileset. This will be installed by default when you select to install the IBM HTTP Server (IHS) as part of the WebSphere V4.0 install process. An equivalent product package exists on the Solaris platform for delivering the IBM SSL module support.

You will also find an example `httpd.conf.sample` configuration file in the Web server conf directory. This can be copied to `httpd.conf` and modified as needed. The comments in the file are self-explanatory and offer specific advice for enabling SSL.

In the example that follows, we manually edit the `httpd.conf` file to support SSL and configure a `VirtualHost` listening on the default HTTPS port of 443. The example was performed on the AIX platform with the IBM HTTP Server (IHS) version 1.3.19, as shipped with WebSphere V4.0.1.

To enable SSL for the IBM HTTP Server (IHS), do the following:

1. Make sure that the IBM HTTP Server and IBM HTTP Administration Server are stopped (we will not be using the Administration Server, as we will be manually editing the `httpd.conf` configuration file directly).
2. Change to the IBM HTTP Server *conf* directory and copy the **`httpd.conf.sample`** file to **`httpd.conf`**.

Note: This task assumes that you have not yet configured your Web server in any way. If you have, you should back up your existing `httpd.conf` file. You will also have to incorporate your previous modifications into the `httpd.conf` once you are finished with the SSL specific steps.

- Using your favorite editor, open the new `httpd.conf`. At a minimum, you should ensure that you have set the `ServerName` directive. In our example this was set to: `ServerName rs617001.itso.ral.ibm.com`.
- To enable 128-bit SSL support on Unix, uncomment the following modules under the Dynamic Shared Object (DSO) Support section:

```
LoadModule ibm_ssl_module      libexec/mod_ibm_ssl_128.so
AddModule mod_ibm_ssl.c
```

- The IBM HTTP Server (IHS) will also be used to service standard HTTP connections on the TCP/IP port 80 default. For this reason, ensure that the following two directives are present and uncommented:

```
Port 80
Listen 80
```

- To allow the IBM HTTP Server (IHS) to respond to requests on more than one address or port, you must create a `VirtualHost` entry. We will use this entry to service encrypted HTTPS requests on the TCP/IP port 443 default. The complete stanza for the `VirtualHost` used in our example is documented below. The `SSLEnable` directive is specified within `VirtualHost` tags to limit the scope of the SSL support. If your only have a single `VirtualHost`, you do not need to specify the `SSLServerCert` option. However, the directive can be used to associate a unique SSL sever certificate label with a specific `VirtualHost`:

```
Listen 443
<VirtualHost rs617001.itso.ral.ibm.com:443>
ServerName rs617001.itso.ral.ibm.com
ErrorLog logs/rs617001443error_log
TransferLog logs/rs617001443access_log
SSLEnable
SSLServerCert WebServer
</VirtualHost>
```

You are not restricted to a single `VirtualHost` entry in the `httpd.conf` file. You may for example wish to configure multiple `VirtualHosts` associated with different `servername` aliases. You should, however, take care when planning your `VirtualHost` configuration, as you are not permitted to have multiple `VirtualHosts` on the same TCP/IP port when each `servername` is an alias of the same IP address. The `NameVirtualHost` directive exists for this task.

If you use multiple `VirtualHosts` with different `SSLServerCert` labels to differentiate between SSL server certificates and find that the same certificate is used for each `VirtualHost`, consider creating an IP alias on your Network Interface Card (NIC) and binding the additional `VirtualHost` `servername` to that IP address. Repeat as necessary to avoid overlapping.

7. After the last VirtualHost stanza, ensure that you disable SSL. This is a precautionary measure to safeguard against SSL being used globally. The IBM HTTP Server (IHS) references multiple SSL server certificates only when stored in a single certificate keyring file. For this reason, the certificate keyfile directive is specified outside of any VirtualHost stanza. Similarly, the SSL timeout values are specified for all VirtualHosts. On timing out, the client is forced to perform another SSL handshake. The settings used as per our example are:

```
SSLDisable
Keyfile /usr/HTTPServer/conf/IHS1319Certs.kdb
SSLV2Timeout 100
SSLV3Timeout 1000
```

8. Save and close your modified httpd.conf file. You can check the syntax of the file with the apachectl command, as follows:

```
# ./apachectl configtest
```

Syntax OK

9. At this point, you can attempt to check the configuration with a Web browser of your choice. If you experience problems, consult the IBM HTTP Server (IHS) error logs, as the error messages are most intuitive.
10. If you copied over the httpd.conf.samples example after installing WebSphere, the now modified httpd.conf will not contain the WebSphere specific directives. In this case, you have two choices; you can either extract the directives from the original httpd.conf that you backed up, or you can re-run the WebSphere install program and choose to re-install the Web server plug-in only, a second time. When prompted, you can select the now SSL enabled **httpd.conf** file as the target for the plug-in.
11. As such, you should ensure that the following entries are present in the httpd.conf file (the paths may be different between OS platforms):

```
LoadModule ibm_app_server_http_module
/usr/WebSphere/AppServer/bin/mod_ibm_app_server_http.so

WebSpherePluginConfig /usr/WebSphere/AppServer/config/plugin-cfg.xml

AddModule mod_app_server_http.c
```

12. Finally, ensure that you update the WebSphere Virtual Host listing to reference the newly created IBM HTTP Server (IHS) VirtualHost identity. Failing to do so will prevent WebSphere from servicing requests on the now secured HTTPS port. In our example, we added the alias:
rs617001.itso.ra1.ibm.com:443.

To the already listed entries:

```
s617001.itso.ra1.ibm.com:80
rs617001.itso.ra1.ibm.com:9080
```

11.2.3 IBM HTTP Server (IHS) Cipher Support Strength

To ensure that your IBM HTTP Server (IHS) operates at the highest possible level of security, you should fully understand how to restrict SSL connections based on encryption cipher strength.

The IBM HTTP Server (IHS) version 1.3.19 supports the following SSL V2 and SSL V3 Ciphers when (mod_ibm_ssl_128.so), the 128-bit IBM SSL Module is loaded (this also includes support for the Triple-DES 168-bit algorithm):

SSL Version 2

27	SSL_DES_192_EDE3_CBC_WITH_MD5	3-DES (168 bit)
21	SSL_RC4_128_WITH_MD5	RC4 (128 bit)
23	SSL_RC2_CBC_128_CBC_WITH_MD5	RC2 (128 bit)
26	SSL_DES_64_CBC_WITH_MD5	DES (56 bit)
22	SSL_RC4_128_EXPORT40_WITH_MD5	RC4 (40 bit)
24	SSL_RC2_CBC_128_CBC_EXPORT40_WITH_MD5	RC2 (40 bit)

SSL Version 3

3A	SSL_RSA_WITH_3DES_EDE_CBC_SHA	3-DES SHA (168 bit)
35	SSL_RSA_WITH_RC4_128_SHA	RC4 SHA (128 bit)
34	SSL_RSA_WITH_RC4_128_MD5	RC4 MD5 (128 bit)
39	SSL_RSA_WITH_DES_CBC_SHA	DES SHA (56 bit)
33	SSL_RSA_EXPORT_WITH_RC4_40_MD5	RC4 MD5 (40 bit)
36	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	RC2 MD5 (40 bit)
32	SSL_RSA_WITH_NULL_SHA	
31	SSL_RSA_WITH_NULL_MD5	
30	SSL_NULL_WITH_NULL_NULL	

TLS Version 1 and SSL Version 3

62	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA	RC4 SHA(56 Bit)
64	TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	DES SHA(56 Bit)

Under each Version heading (SSL V2, SSL V3, TLS V1), the strongest cipher is listed first, descending to the weakest.

Note: In a previous IHS release, the cipher specification was incorrectly documented:

```
33|SSL_RSA_EXPORT_WITH_RC4_40_MD5|RC4 SHA (128 bit)
35|SSL_RSA_WITH_RC4_128_SHA|RC4 SHA (40 bit)
```

The correct listing should have read:

```
33|SSL_RSA_EXPORT_WITH_RC4_40_MD5|RC4 MD5 (40 bit)
35|SSL_RSA_WITH_RC4_128_SHA|RC4 SHA (128 bit)
```

Using Netscape to check the integrity of your SSLCipherSpec

You can use the following steps to verify the SSL encryption level set by the IBM HTTP Server (IHS).

1. Modify an SSL enabled VirtualHost entity in your httpd.conf configuration file to use a specific encryption cipher. In this example, we will use the highest level of encryption possible, through the Triple-DES SHA (168-bit) cipher suite. This is achieved by adding the SSLCipherSpec 3A (3A is the short name for the full cipher algorithm) directive to the VirtualHost entry, as shown below:

```
<VirtualHost rs617001.itso.ral.ibm.com:443>
ServerName rs617001.itso.ral.ibm.com
ErrorLog logs/rs617001443error_log
TransferLog logs/rs617001443access_log
SSLEnable
SSLServerCert WebServer
SSLCipherSpec 3A
</VirtualHost>
```

2. Stop and start the IBM HTTP Server (IHS) to ensure that the SSLCipherSpec modification is included in the runtime.
3. Using Netscape Communicator 4.7, select the **Configure SSL V3** option found under the Security Info management window, as shown in Figure 11-17. By default, Netscape Communicator 4.7 does not ship with 128-bit cryptography support. This functionality can be achieved by installing the Fortify (found at <http://www.fortify.net>) encryption package for Netscape Communicator.
From the Netscape menu, select: **Communicator -> Tools -> Security Info -> Navigator -> Configure SSL V3.**

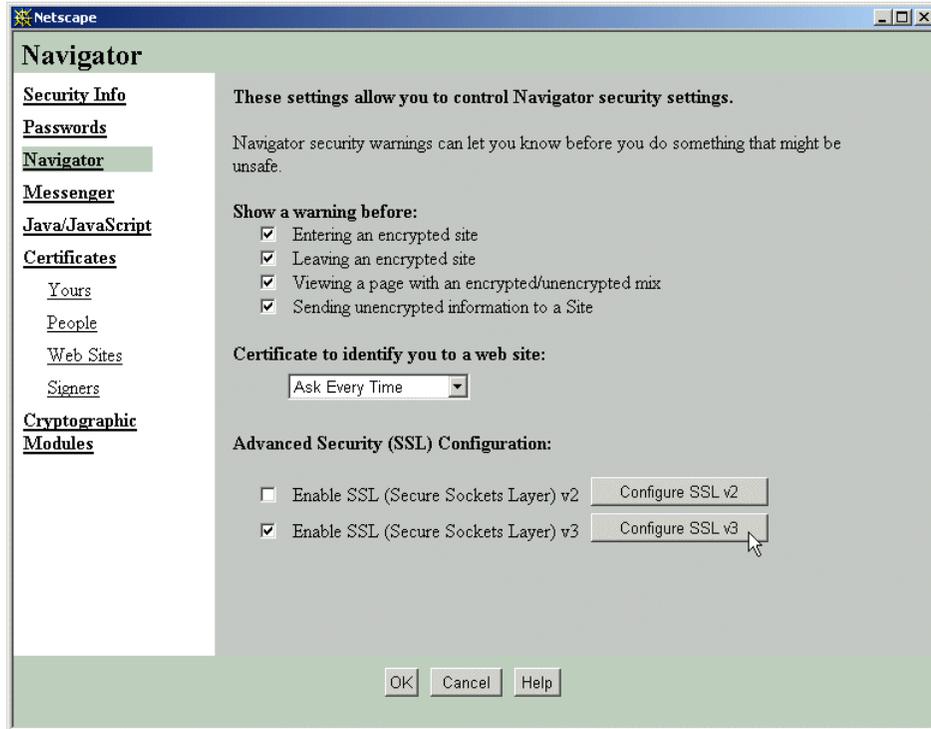


Figure 11-17 Netscape Security Information management

4. Selecting the **Configure SSL V3** option will launch the Configure Ciphers window, as shown below in Figure 11-18. Depending on what you find, the settings will enable or restrict your use of certain encryption algorithms.

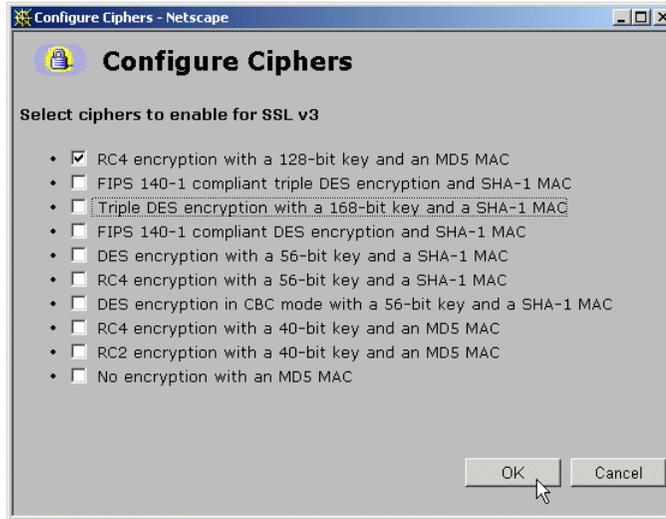


Figure 11-18 Configuring ciphers for Netscape Navigator

5. By deselecting every option apart from the **RC4 encryption with a 128-bit and a MD5 MAC** entry, you will restrict the SSL handshaking achievable by the Netscape browser. Only (RC4 128-bit) HTTPS connections will be possible. In certain circumstances, this might be a beneficial practice, as you might wish to safeguard against using any of the low-grade encryption algorithms (40-bit or 56-bit).
6. If you now attempt to retrieve an SSL secured resource served by the IBM HTTP Server (IHS) VirtualHost entry, Netscape will prompt you with the message shown in Figure 11-19. You will not be able to establish an SSL/HTTPS connection with the IBM HTTPServer (IHS), as the server only supports Triple-DES SHA (168-bit) communication.



Figure 11-19 Netscape - Security error message

7. You may specify more than one SSLCipherSpec directive per VirtualHost to extend support to multiple encryption algorithms. This way, if a client Web browser does not support the first encryption algorithm, it will attempt to establish an SSL connection using the next, weaker algorithm.

```
<VirtualHost rs617001.itso.ral.ibm.com:443>
  ServerName rs617001.itso.ral.ibm.com
  ErrorLog logs/rs617001443error_log
  TransferLog logs/rs617001443access_log
  SSLEnable
  SSLServerCert WebServer
  SSLCipherSpec 3A
  SSLCipherSpec 35
  SSLCipherSpec 34
</VirtualHost>
```

Important: All available cipher specifications are enabled by default, when no SSLCipherSpec directive is present.

8. Figure 11-20 shows the cipher strength information for a secure Triple-DES SHA (168-bit) HTTPS connection established between Netscape Communicator and the IBM HTTP Server (IHS).

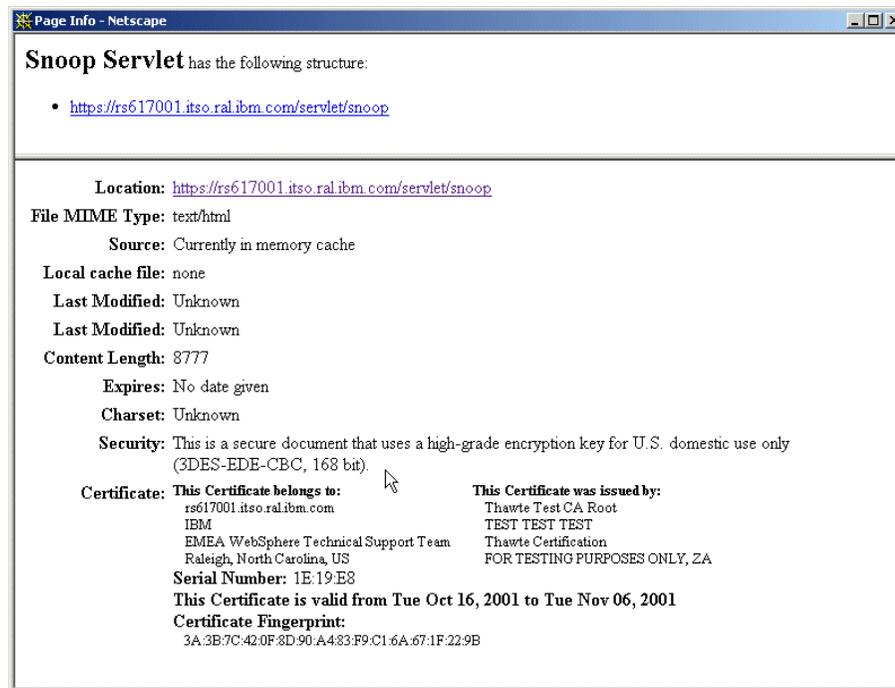


Figure 11-20 Netscape Navigator - Page Information

9. The Microsoft Internet Explorer Properties window, shown in Figure 11-21, is not so explanatory. Nevertheless, you can determine the cipher strength used in the HTTPS connection.

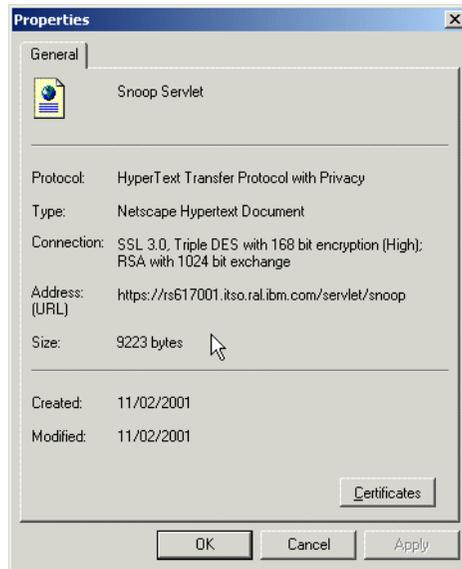


Figure 11-21 Microsoft Internet Explorer - Page Properties

11.3 Client-Side Certificates for Authentication

Authentication can be extended to an individual or entity with the use of a personal digital certificate. When resident in a Web browser, such a certificate can be used to ensure the legitimacy claimed by that individual. That is, the certificate acts to guarantee the client's claim as to their identity.

The steps necessary to configure WebSphere to support Client-Side Certificate authentication are documented in this section. In addition, we briefly introduce the concept of Public Key Infrastructure (PKI) by outsourcing the generation of certificates to a third-party certificate authority (CA), albeit using a trial PKI program offered by a CA.

The assumption is made that you have previously enabled WebSphere Global Security using the Lightweight Third Party Authentication (LTPA) option, against a remote LDAP Directory Server.

The following products were used in the configuration documented:

- ▶ AIX 4.3.3.0-0.8
- ▶ IBM SecureWay Directory Server 3.2.1.0 for AIX
- ▶ IBM DB2 Universal Database 7.2.1 for AIX
- ▶ IBM HTTP Server 1.3.19 for AIX
- ▶ Microsoft Internet Explorer V6.00.2462.0000 / Cipher Strength128-bit for Windows
- ▶ Netscape Communicator 4.7 for AIX and Windows

11.3.1 Securing a Web Application to use client certificates

To enable client side certificate-based authentication, you must modify the authentication method defined on the J2EE Web Module that you wish to manage. It might be that the Web Module has already been configured to use the basic challenge authentication method. In this case, you will simply need to modify the challenge type to *client cert*. As such, this functionality is delivered to the WebSphere administrator in the Application Assembly Tool (AAT). However, it is envisaged that developers will use the WebSphere Studio Application Development (WSAD) environment to achieve the same result.

1. Launch the WebSphere Application Assembly Tool (AAT).

This step can be undertaken either before an enterprise application archive .ear file has been deployed into WebSphere or after deployment into WebSphere, although the latter option is discouraged in a production environment, as it involves opening the expanded archive correlating to the enterprise application archive in question, found under the WebSphere installedApps directory.

2. Locate and expand the Web Module package under the Application for which you wish to enable the client side certificate authentication method.
3. Select the appropriate Web Application, switch to the Advanced tab and modify the Authentication method to read **Client cert**. The Realm name is the scope of the login operation and should be same for all participating resources.
4. Click **OK** and then save the changes made with Application Assembly Tool (AAT).
5. If the modification was made to a resource already deployed into WebSphere, stop and start the associated Application Server containing the resource, so that the security modification can be included in the runtime.

11.3.2 Obtaining a personal certificate

One method for obtaining a personal certificate is through a Public Key Infrastructure (PKI) program offered by a third-party Certificate Authority (CA). Such programs are geared towards creating multiple personal certificates quickly and easily, with numerous value added functions offered by each CA.

Alternatively, you can acquire the IBM Tivoli SecureWay PKI package or a similar product from another vendor and implement your own PKI solution. In this case, you will be tasked with the overhead of managing the PKI infrastructure, as well as that of creating the individual certificates for each authenticating user.

Generating and installing a PKI personal certificate

For demonstration purposes, we opted to use the free Personal Certificate Program offered by Thawte Consulting, available at <http://www.thawte.com>. Other Certificate Authorities (CAs) offer similar programs and should not be disregarded. We will later use this certificate to demonstrate client side certificate authentication with WebSphere Application Server.

The following is provided only as a rough guide, without warranty, and may be subject to change at Thawte Consulting's discretion. In this case, you should consult the official online documentation for the definitive configuration procedures endorsed by the CA.

1. Start the Web browser of your choice and connect to the following URL:
<http://www.thawte.com>
2. Join the free Personal Certificate Program and complete the necessary registrations steps.
3. Request a certificate using the X.509 format, selecting the target Web browser of your choice where the certificate will be installed.
4. Click the **get X.509 Certificate** button.
5. Click **Next** in the Employment window.
6. Ensure your e-mail address from the registration step is selected, as it will form part of the certificate Common Name (CN).
7. Click **Next** at the Strong Extranet Identities prompt.
8. Accept the Default Extensions.
9. Select **Microsoft Enhanced Cryptographic Provider V1.0** as determined automatically by your Web browser.
10. Clicking **Next** will launch a new window from your Web browser entitled Creating a new RSA exchange key; select the **Medium** security level.
11. Finally, confirm the certificate request and select **Finish**.

The resulting certificate will contain a Distinguished Name (DN) including the Common Name (CN) *Thawte Freemail Member* and your e-mail address, as specified during registration. The inclusion of the reference to the Thawte Freemail Member is a restriction placed on the free program offering.

It is envisaged that in a production environment, the PKI solution that you choose to implement will allow you to specify your own fully qualified Distinguished Name (DN).

12. Proceed to the Track Your Certificate Status window.
13. When the status of the certificate changes from Pending to issued, you can install the certificate by clicking the **Fetch This Certificate** and **Install this Cert** buttons, respectively.

Supplement these steps if you opted to use an alternative Certificate Authority (CA) PKI program.

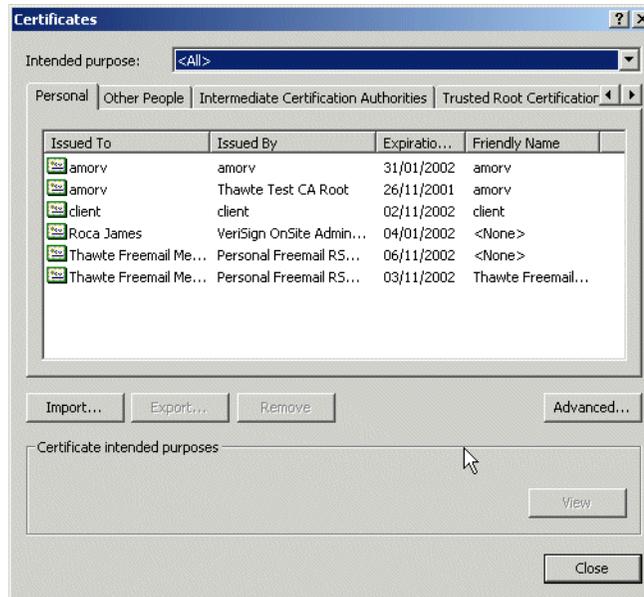


Figure 11-22 Microsoft Internet Explorer - installed certificates

Microsoft Internet Explorer users can then check the installation of personal certificates by opening the certificates window found under the Explorer **Tools -> Internet Options -> Content** menus, as shown above in Figure 11-22.

Netscape Communicator similarly allows users to manage personal certificates by selecting the following menus: **Communicator -> Tools -> Security Info -> Certificates -> Yours**.

The *WebSphere V4.0 Handbook* (SG24-6176) provides further documentation on importing and managing personal certificates with Microsoft Internet Explorer.

The following checks are also recommended on any certificate installed into Microsoft Internet Explorer, for use as a client side certificate. Double-click any certificate entry and verify the following:

1. Under the General tab, the certificate's intended use includes:
Proving your identity to a remote computer (required).
2. Under the General tab:
You have a private key that corresponds to this certificate (required).
3. Under the Certificate Path, the Certificate status is given:

If you are presented with the message: This CA Root certificate is not trusted because it is not in the Trusted Root Certification Authorities Store, then you must install the corresponding signing root CA certificate in the Certification Authorities Store.

Figure 11-23 below details the resulting personal certificate generated when using the Thawte Freemail Certificate Program. The certificate subject Distinguished Name (DN) includes two components; the e-mail entity (E) rocaj@uk.ibm.com and the Common Name (CN) Thawte Freemail Member.

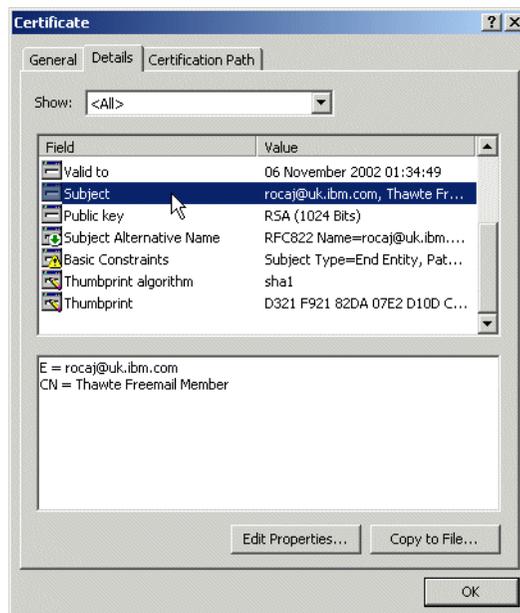


Figure 11-23 Microsoft Internet Explorer - certificate properties

This concludes the steps necessary to generate and install a personal client side certificate into Microsoft Internet Explorer.

11.3.3 LDAP advanced security settings

Certificate-based authentication requires either that WebSphere map the entire certificate subject Distinguished Name (DN) to a like LDAP Distinguished Name (in this case, the LDAP Distinguished Name (DN) formed in part from the LDAP Directory hierarchy must match element for element the certificate subject Distinguished Name), or that WebSphere certificate filtering be used to map a certificate subject Distinguished Name to a specific LDAP field, for a given LDAP user.

As structure and hierarchy are of concern when managing an LDAP Directory, it is not always possible to use the same Distinguished Name (DN) that is supported by the client side certificates. For example, if you have ever attempted to create a Web server certificate signed by a third-party Certificate Authority (CA), you will know that the Common Name (CN) needs to include the fully qualified host name, in our example: CN=rs617001.itso.ra1.ibm.com.

We acknowledge that this is more an issue having to do with Web server certificates and anticipate that the restriction is not of any concern when dealing with personal client side certificates.

Likewise, the LDAP Directory Distinguished Name (DN) or suffix does not necessarily have to conform to the O=company,C=us standard, as shown in many an example. Indeed, the fictitious Distinguished Name (DN) used in Section 11.6, “Securing WebSphere LTPA with SSL” on page 282 specifies the following Distinguished Name (DN): OU=uk,DC=internetchaos,DC=com.

If you have used the IBM ikeyman tool, you will be aware that attributes permitted in the certificate subject Distinguished Name (DN) only include the following:

```
CN=commonName
OU=organizationUnit
O=organizationName
L=localityName
S=stateName
C=country
```

Obviously, the IBM ikeyman tool was never designed to create personal client side certificates, with the flexibility to specify a Distinguished Name (DN) matching that of an LDAP Directory Server. This problem is overcome if we use the WebSphere certificate filtering option.

For completeness, we will demonstrate both mapping by exact Distinguished Name (DN) and filtered certificate attribute mapping.

Using the WebSphere LDAP Certificate Filter option

This section assumes that you have successfully installed a personal certificate into a client Web browser and that you have previously enabled WebSphere Global Security authenticating users against a remote LDAP Directory Server. It is anticipated that the personal certificate subject Distinguished Name (DN) does not necessarily match, in any way, your LDAP Distinguished Name (DN).

If you generated a certificate using Thawte's free Personal Certificate Program, then your certificate will hold the characteristics similar to those shown in Example 11-11. Here, the Owner attribute equates the certificate *subjectDN*, and, as in the case of our example, uniquely contains the value: EmailAddress=rocaj@uk.ibm.com, CN=Thawte Freemail Member.

If you used an alternative PKI solution, the *subjectDN* will be different, but equally unique, with the Issuer (signer) value being different again.

Example 11-11 Personal Freemail certificate

```
Alias name: 461d2384251e922f08231fea41a451ce_864afe53-5970-428b-816a-2980
Creation date: Fri Nov 02 17:03:35 EST 2001
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: EmailAddress=rocaj@uk.ibm.com, CN=Thawte Freemail Member
Issuer: CN=Personal Freemail RSA 2000.8.30, OU=Certificate Services, O=Thawte,
L=Cape Town, ST=Western Cape, C=ZA
Serial number: 602ec
Valid from: Fri Nov 02 16:57:25 EST 2001 until: Sat Nov 02 16:57:25 EST 2002
Certificate fingerprints:
    MD5: 00:08:A5:AE:77:02:EE:27:E1:45:76:57:12:7F:9E:B8
    SHA1: 75:1F:8A:87:72:FB:4A:0E:06:59:94:6E:75:C3:1B:EA:0B:5D:A9:9A
```

The next step is to modify WebSphere LDAP filtering rules to map the certificate subjectDN field to the IBM SecureWay LDAP uniqueIdentifier field for a given user. You do not necessarily have to use the SecureWay LDAP uniqueIdentifier field. However, you should ensure that the data type of the field selected is capable of handling the specific value. Also ensure that WebSphere has the right to search such a field when performing authentication. As such, we will set the values to:

Certificate Mapping: Certificate Filter

Certificate Filter: uniqueIdentifier=\${SubjectDN}

Ensure that the certificate attribute selected for authentication is unique between certificates. For example, performing a search on the certificate signature algorithm name is not advisable.

Complete the following steps to configure WebSphere V4.0 to use LDAP certificate filtering:

1. Launch the WebSphere Security Center from the WebSphere Administration Console: **Console -> Security Center**.
2. Select the **Authentication** tab and click the **Advanced** settings button.
3. Ensure that the Certificate Mapping field is set to Certificate Filter and that the Certificate Filter reads `uniqueIdentifier=${SubjectDN}`, as shown in Figure 11-24 below:



The screenshot shows a dialog box for LDAP settings. The fields are as follows:

* Initial JNDI Context Factory:	com.sun.jndi.ldap.LdapCtxFactory
User Filter:	(&(uid=%*)(objectclass=ePerson))
Group Filter:	(objectclass=groupOfUniqueNames))
User ID Map:	*:uid
Group ID Map:	*:cn
Group Member ID Map:	groupOfUniqueNames:uniqueMember
Certificate Mapping:	Certificate Filter
Certificate Filter:	uniqueIdentifier=\${SubjectDN}

Buttons: OK, Cancel, Help

Figure 11-24 Advanced LDAP settings

4. Click **OK** to save the changes.

Note: Modifying any of the attributes in the LDAP Advanced Properties window will cause the Directory Type to read Custom, rather than your selected directory type on the main Authentication tab.

5. Back on the Authentication tab, click **Apply**.
Consult Section 11.6, “Securing WebSphere LTPA with SSL” on page 282 to view the parameters used in the LDAP Settings window. You must complete the Bind Distinguished Name and Bind Password fields to query the LDAP Directory server of your choice as an authenticated user.
6. Stop and start the WebSphere Administrative Server to implement the advanced LDAP modifications.

Modifying the SecureWay LDAP uniqueIdentifier field

After you have modified the WebSphere V4.0 LDAP Certificate Filter setting, you must update your selected Directory Server to contain the entry that will match and authenticate the end user. In our case, we must add the following string to the *uniqueIdentifier* field of a specific LDAP user:

EmailAddress=rocaj@uk.ibm.com, CN=Thawte Freemail Member.

To modify a user in the IBM SecureWay LDAP Directory registry, do the following:

1. Launch the SecureWay Directory Management Tool (DMT).
2. Rebind as an Authenticated User with adequate privileges to modify user credentials.
3. Expand the Directory tree and select the user entity against which you wish to authenticate the personal client certificate. This is shown in Figure 11-25.

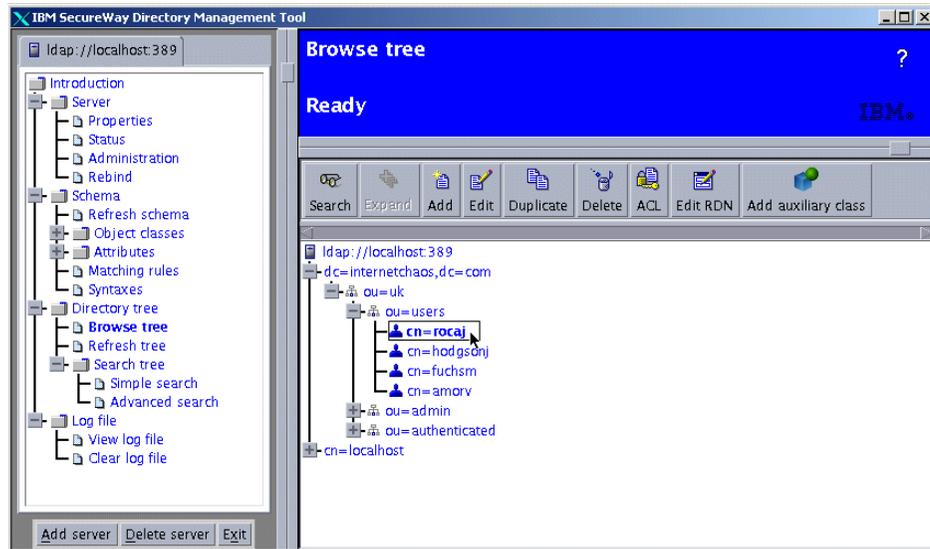


Figure 11-25 Directory Management Tool - Browse Tree

4. Double-click the selected user you wish to update. This will launch a new window allowing you to modify user specific attributes.
5. Select the **Other** tab and find the *uniqueIdentifier* field, then enter:
EmailAddress=rocaj@uk.ibm.com, CN=Thawte Freemail Member or a similar value as defined in your certificate subjectDN. This is shown in Figure 11-26.

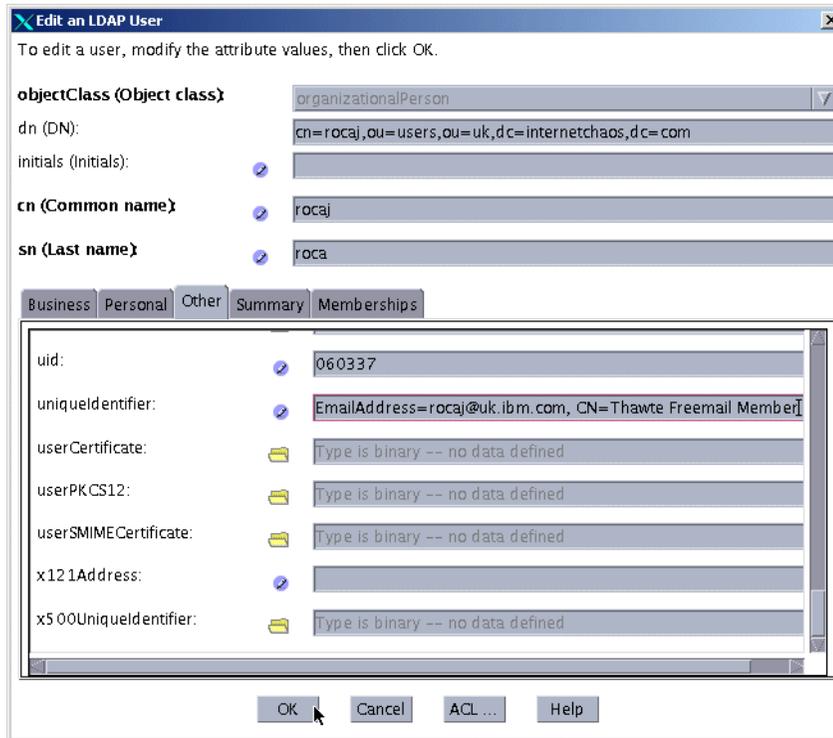


Figure 11-26 LDAP user entry details

6. Click **OK** when you are done.
7. Repeat this step for each individual user against which you wish to perform client side certificate authentication, setting the appropriate certificate string in the *uniqueIdentifier* field each time.

Modifying the Web server to support client certificates

Finally, you must ensure that the selected Web server is configured to request client side certificates. In the case of the IBM HTTP Server, you must edit `httpd.conf` and include the `SSLClientAuth` directive, specifying the required option. This should be specified for each `VirtualHost` entry that you wish to secure with client side certificate authentication.

```
<VirtualHost rs617001.itso.ral.ibm.com:443>
  ServerName rs617001.itso.ral.ibm.com
  ErrorLog logs/rs617001443error_log
  TransferLog logs/rs617001443access_log
```

```
SSLEnable
SSLServerCert WebServer
SSLClientAuth required
</VirtualHost>
```

Ensure that you stop and start the Web server to incorporate the change into the runtime. Failing to do so will mean that client certificates are not requested.

Testing your client side certificate

We suggest that you refrain from modifying the Default Application sample to use the client certificate authentication method. This way, you can use the *snoop* servlet, referenced by `https://<your_server_name>/servlet/snoop`, to determine if your browser is correctly passing a client certificate.

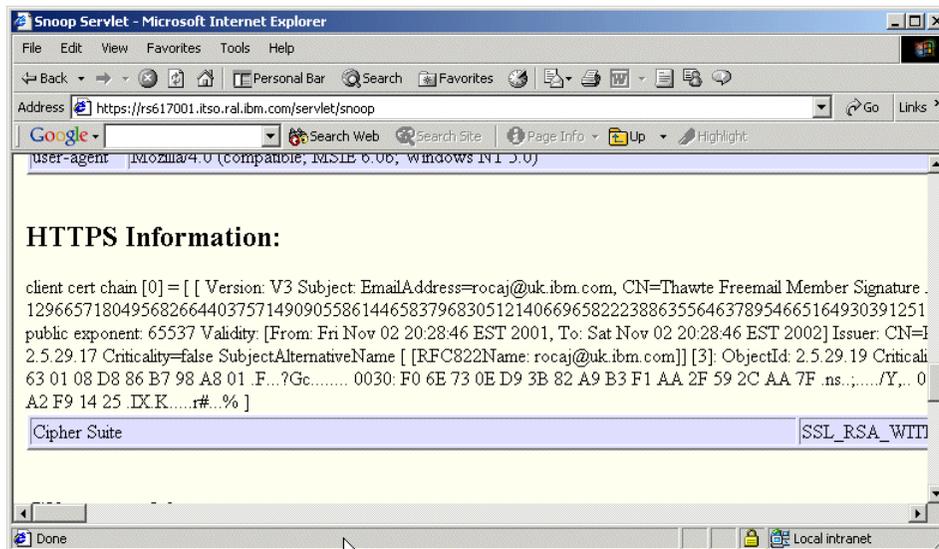


Figure 11-27 Testing the client certificate, using the *snoop* servlet

In Figure 11-27 above, the personal certificate installed in Microsoft Internet Explorer has successfully been passed to WebSphere. In the case that a Client fails to pass a certificate, WebSphere will only return the Cipher Suite specification as used in the HTTPS connection. If this occurs, check that the client browser has a valid certificate installed and that your chosen Web server is set to request client certificates. With the IBM HTTP Server (IHS), this is achieved with the inclusion of the `SSLClientAuth` directive in the `httpd.conf` file.

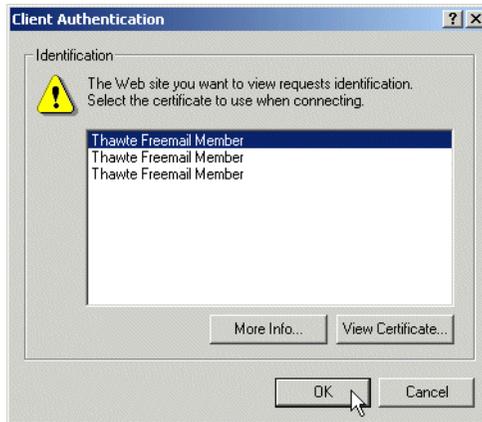


Figure 11-28 Microsoft Internet Explorer - selecting client certificate for authentication

A correctly configured implementation will prompt the client Web browser user to select a personal certificate when accessing a resource protected by the SSLClientAuth directive; this is shown above in Figure 11-28 for Microsoft Internet Explorer. Netscape Navigator/Communicator users are prompted in a similar fashion to select the personal certificate of their choice. In both cases, Microsoft and Netscape allow the personal certificates in each respective browser to be protected with a password. This further protects the certificate against unauthorized submission.

Using the exact Distinguished Name

In contrast to the WebSphere LDAP certificate filtering option, this method depends on mapping the certificate subject Distinguished Name (DN) to an exact entry in your chosen LDAP registry. In this case, the LDAP Distinguished Name (DN) is constructed from the Relative Distinguished Name (RDN) of a specific user and the concatenated hierarchal parents of the LDAP topology tree.

Although the mapping is case-insensitive between the certificate SubjectDN and the LDAP DN. One restriction that you might encounter is that of being unable to create a certificate matching your LDAP DN. For example, the LDAP structure used in Section 11.6, “Securing WebSphere LTPA with SSL” on page 282 uses a DN to represent the fictitious domain name of internetchaos.com. In this case, an individual LDAP entry looks like:
CN=rocaj,OU=users,OU=uk,DC=internetchaos,DC=com.

If you have experimented with ikeyman, you will be aware that you can only create certificates with a predefined set of attributes for the SubjectDN. A typical example may be: CN=rs617001.itso.ra1.ibm.com, OU=IBM, O=International Technical Support Organization, L=Raleigh, ST=North Carolina, C=US.

This corresponds to the permitted fields that the IBM ikeyman tool supports:

```
CN=commonName
OU=organizationUnit
O=organizationName
L=localityName
S=stateName
C=country
```

The Java command line keytool utility, that ships with WebSphere and Java 1.3, will, however, allow you to specify your own custom fully qualified Distinguished Name (DN) for use with client side certificates. This is probably not of any concern if you are using a fully functional PKI solution, as you will more than likely be able to specify your own fully qualified certificate subject Distinguished Name (DN) using the supplied software.

In the example that follows, we document the steps necessary to create two certificates that match two different users in our SecureWay LDAP registry.

1. Set the environment to include the PATH for the Java keytool executable, if it is not already set. Then change to a directory where the resulting certificates will be created. Windows users should supplement these steps accordingly.

```
#export PATH=$PATH:/usr/WebSphere/AppServer/java/jre/bin/
#cd /export/home/certificates
```

2. The two LDAP Distinguished Names (DNs) that we will match are:

```
cn=rocaj,ou=users,ou=uk,dc=internetchaos,dc=com
cn=amorv,ou=users,ou=uk,dc=internetchaos,dc=com
```

3. Using the Java keytool utility, create a private/public self-signed certificate key pair associated with the first user `cn=rocaj`. Supplement your own values as required. Specify `RSA` for the private key to ensure that the `MD5withRSA` signature algorithm is used, as not all Web browsers support the `DSA` cryptograph algorithm (which is the default when `RSA` is not specified). You need to set a password of at least six characters to protect the private key. Finally, the `keystore` and `keystore (storepass)` password are specified.

Note: In the examples that follow in this section, a carriage return is only required after the last keystroke / word.

Example 11-12 Creating a certificate- 1

```
#keytool -genkey -keyalg RSA -dname  
"cn=rocaj,ou=users,ou=uk,dc=internetchaos,dc=com" -alias rocaj -keypass  
websphere -keystore testkeyring.jks -storepass websphere
```

4. Create the second private/public self-signed certificate key pair in the same manner for the user `cn=amorv`.

Example 11-13 Creating a certificate- 2

```
#keytool -genkey -keyalg RSA -dname  
"cn=amorv,ou=users,ou=uk,dc=internetchaos,dc=com" -alias amorv -keypass  
websphere -keystore testkeyring.jks -storepass websphere
```

5. At this point, the keystore `testkeyring.jks` will contain two self-signed certificates, with the Owner (entity) being the same as the Issuer (signer) for each certificate.
6. Next, to ensure the integrity and authenticity of the certificates, we need to get each certificate signed by the Certificate Authority (CA). If you are implementing your own PKI solution, you do not necessarily need to submit the Certificate Signing Request (CSR) to a third-party CA. You do, however, need to ensure that any client Web browsers receiving the certificate have the appropriate authenticating root certificate installed as a trusted authority. The following two examples extract the Certificate Signing Request (CSR) for the two user certificates that we are creating.

Example 11-14 Generating the CSR- 1

```
#keytool -v -certreq -alias rocaj -file rocajReq.csr -keypass websphere  
-keystore testkeyring.jks -storepass websphere
```

And for the second user identified by `cn=amorv`.

Example 11-15 Generating the CSR- 2

```
#keytool -v -certreq -alias amorv -file amorvReq.csr -keypass websphere  
-keystore testkeyring.jks -storepass websphere
```

7. We used the free Test SSL certificate program offered by Thawte Consulting (found at <http://www.thawte.com>) to sign our Certificate Signing Requests (CSRs). In each case, we selected the **Custom Cert (configure below)** option and ensured that the Certificate format was set to use default for your kind of certificate. We also selected **Generate an X.509v3 Certificate** and saved the two resulting files as `rocajRes.arm` and `amorvRes.arm`, respectively.
8. Before you can receive the signed certificate response back into the keystore, you must import the Certificate Authority's trusted root certificate into the keystore. Copy and paste the Thawte test root certificate in BASE64-encoded

ASCII data format to a file called ThawteTestCA.arm. Then add the test root CA certificate into the keystore with the following command:

Example 11-16 Adding the test root CA

```
keytool -import -alias "Thawte Test CA Root" -file ThawteTestCA.arm -keystore
testkeyring.jks -storepass websphere
```

9. Import the two certificate responses from the CA into the keystore using the same alias name as first given to the self-signed certificates. In our example, these were rocaj and amorv, respectively. Using an alternative alias name will generate a new signer certificate and not a personal certificate chain.

Example 11-17 Importing the certificate response- 1

```
keytool -import -trustcacerts -alias rocaj -file rocajRec.arm -keystore
testkeyring.jks -storepass websphere
Certificate reply was installed in keystore
```

And for the second user identified by cn=amorv.

Example 11-18 Importing the certificate response- 2

```
keytool -import -trustcacerts -alias amorv -file amorvRec.arm -keystore
testkeyring.jks -storepass websphere
Certificate reply was installed in keystore
```

10. At this point, you need to launch the IBM JSSE ikeyman utility found in the WebSphere *bin* directory (this action is necessary, as the Java keytool command line utility does not support exporting the private certificate key). The IBM JSSE ikeyman tool supports the PKCS12 format and will allow you to export the private key associated with any certificate (the public key is also exported).
11. Open the testkeyring.jks keystore previously created with the Java keytool utility and select the first certificate from the Personal Certificates drop-down menu.
12. Click **Export**, ensure that you select the **PKCS12** format and name the target file accordingly. You will be prompted to specify and confirm the password used to protect the resulting PKCS12 file. We named the two files in our example rocajprivate.p12 and amorvprivate.p12, respectively.
13. Before you can install a personal certificate signed by a third-party Certificate Authority (CA) into a given Web browser, you must make sure that the same root certificate of the authenticating CA is installed as a Trusted Authority in the browser. Such certificates may be one of either two types: an Intermediate Certificate Authority or a Trusted Root Certificate Authority.

Note: The example documented here uses the Thawte Test Root CA certificate to sign the personal private/public certificate key pair. This certificate is not installed by default in either Microsoft Internet Explorer or Netscape Communicator/Navigator. Install the required root certificate by following the instructions on Thawte's Web site.

14. To install either of the personal certificates into Netscape Communicator, simply use the Import a Certificate option found in the Your Certificates window, as shown below in Figure 11-29. The path to selecting the menu is as follows: **Communicator -> Tools -> Security Info -> Certificates -> Yours.**

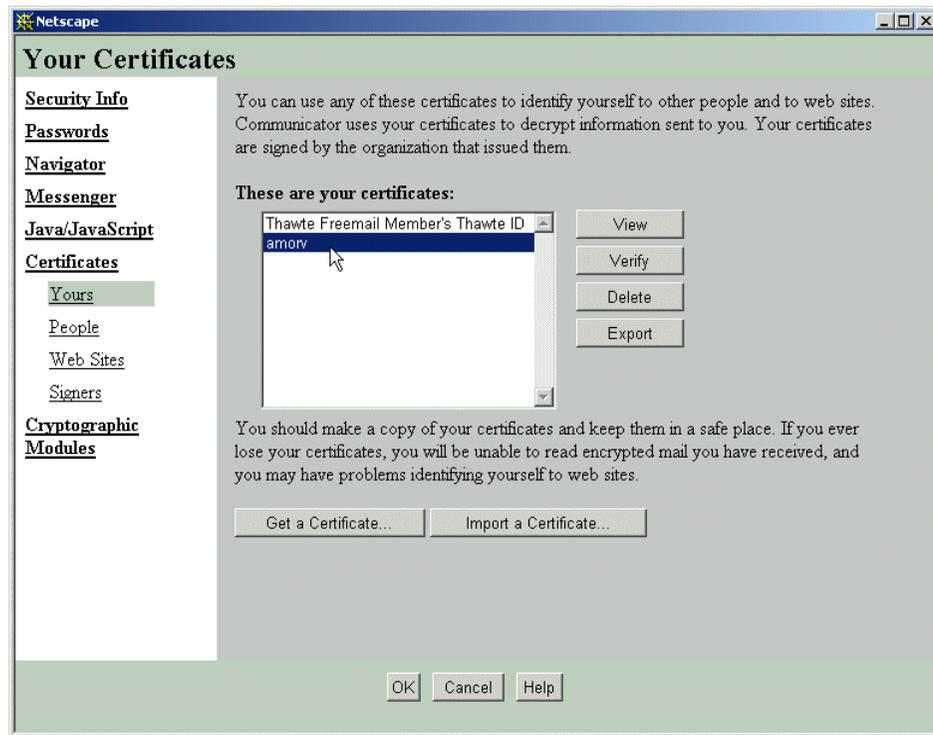


Figure 11-29 Certificates under Microsoft Internet Explorer

15. Netscape will prompt you to enter a password or PIN for the Communicator Certificate DB when you attempt to import the certificate. You should enter the password as used when first initializing your Certificate DB.

If you have not previously used personal certificates with Netscape, you will be prompted to complete some additional steps to guard against the unauthorized submission of your personal certificates.

16. You are also prompted to enter the password protecting the PKCS#12 certificate file, as set when you exported the personal private/public certificate key pair in ikeyman.
17. Once imported, select the **Verify** option to check integrity and validity of the certificate. If you failed to install the root Certificate Authority (CA) certificate, your certificate will fail the verification.

Ensure that you have modified your chosen Web server to support client side certificate requests, as documented previously in this chapter. Assuming that you have also refrained from implementing client certificate authentication on the Default Application sample, you can at this point check that the personal certificate is passed to WebSphere by using the snoop servlet. In this case, you will be prompted to select a personal certificate when you connect to the following URL: `https://<your_server_name>/servlet/snoop`.

A correctly configured implementation will prompt the Web browser user to select a personal certificate when accessing a resource protected by the SSLClientAuth directive. This is shown in below in Figure 11-30.

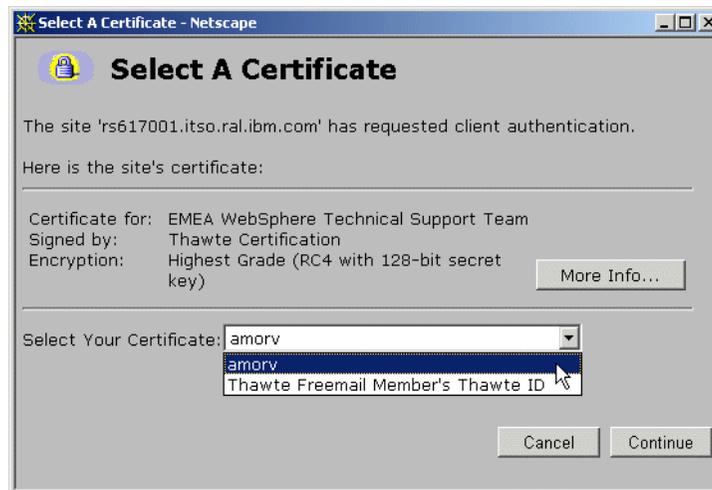


Figure 11-30 Prompt for certificate window

If you check the HTTPS Information displayed by the snoop servlet, you should now see the certificate SubjectDN matching the following: Subject: CN=amorv, OU=users, OU=uk, DC=internetchaos, DC=com.

Configuring WebSphere for use with exact mapping

Complete the following steps to ensure that WebSphere is set to use the exact Distinguished Name (DN) mapping option:

1. Launch the WebSphere Security Center from the WebSphere Administration Console: **Console -> Security Center**.
2. Select the **Authentication** tab and click the **Advanced** settings button.
3. Ensure that the LDAP Certificate Mapping field is set to Exact Distinguished Name. This option will grey out the Certificate Filter field, so that no filter may be specified.
4. Click **OK** for the changes to be saved.
5. Back on the **Authentication** tab, click **Apply**.
6. Stop and start the WebSphere Administrative Server to implement the advanced LDAP modifications.
7. To test Client side certificate authentication with WebSphere, access an application Web resource that has been secured with the Client Cert authentication method. Recall that we documented this step initially in this section, using the WebSphere Application Assembly Tool (AAT).

11.4 Configuring SSL between Web server and WebSphere Application Server

This section documents the steps necessary for implementing secure HTTPS communication between the Web server plug-in and the embedded HTTP server of a WebSphere Web Container. By default, this connection is not secure, even when WebSphere Global Security is enabled.

The following steps are mandatory for generating the certificates for SSL handshaking/authentication between the two differing peers.

1. Create a self-signed certificate for the Web server plug-in.
2. Create a self-signed certificate for the WebSphere embedded HTTP Server (Web Container).
3. Exchange the public keys.
4. Modify the Web server plugin-cfg.xml to use SSL/HTTPS.
5. Modify the WebSphere embedded HTTP Server (Web Container) to use SSL/HTTPS.

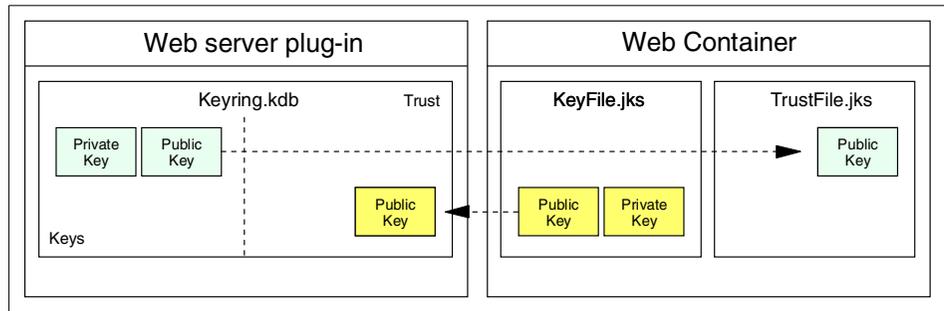


Figure 11-31 Certificates

Figure 11-31 illustrates the exchange of the public certificate keys associated with each peer participating in the secure SSL communication.

11.4.1 Generating a self-signed certificate for the Web server plug-in

The following steps will guide you through the process of generating a self-signed certificate for the Web server plug-in.

1. Create a suitable directory on the Web server host for storing the keyring file referenced by the plug-in and associated files.
2. Launch the IBM ikeyman tool that ships as part of GSKit and supports the CMS key database format. This is not the JSSE ikeyman version that ships under the WebSphere bin directory.

For AIX, ikeyman ships as part of the `http_server.ssl.core.1.3.19.0` fileset and can be invoked from the `/usr/bin` directory.

The Windows equivalent ikeyman version is installed by default as `C:\Program Files\ibm\gsk5\bin\gsk5ikm`.

3. From the ikeyman menu bar, select **Key Database File -> New**.
4. Set the following settings and click **OK** when you are done.

Key database file: CMS Key Database File
 File name: WASV4PlugIn.kdb
 Location: `/usr/HTTPServer/conf/certs` (or the directory of your choice)
5. At the Password Prompt window, enter the password of your choice. The more random the password, the higher the password strength. Finally, stash the password to a file so that the plug-in can use the password to gain access to the certificates contained in the key database.
6. As we are only going to be implementing a peer-to-peer SSL connection between the Web server plug-in and the embedded HTTP server of any given

Web container, we are not concerned with the signer certificates of the publicly circulating root certificate authorities (CAs). In this case, optionally delete all of the CA trusted signer certificates.

7. From the ikeyman menu bar select **Create -> New Self-Signed Certificate** to create a new self-signed certificate key pair. The following options then need to be specified; you may choose to complete all of the remaining fields for the sake of completeness:

Key Label: WASV4Plugin

Version: X509 V3

Key Size: 1024

Common Name: rs617001.itso.ral.ibm.com

Organization: IBM

Country: US

Validity Period: 365

Click **OK** when you are finished.

8. Extract the public self-signed certificate key, as this will be used later by the embedded HTTP server peer to authenticate connections originating from the plug-in.
9. Select **Personal Certificates** in the drop-down menu and select the **WASV4Plugin** certificate that was just created.

10. Click the **Extract Certificate** button, ensuring that **WASV4Plugin** remains selected. Extract the certificate to a file:

Data type: Base64-encoded ASCII data

Certificate file name: WASV4PluginPubCert.arm

Location: /usr/HTTPServer/conf/certs (or the directory of your choice)

Click **OK** when you are finished.

11. Close the key database and quit ikeyman when you are finished.

11.4.2 Generating a self-signed certificate for a Web Container

The following steps will show how to generate a self-signed certificate for the WebSphere Web Container.

1. Launch the IBM JKS capable ikeyman version that ships under the WebSphere bin directory. On Unix systems, this is invoked by running the **ikeyman.sh** script.
2. From the ikeyman menu bar select **Key Database File -> New**.

3. Set the following settings and click **OK** when you are done:

Key database file: JKS

File name: EmbeddedHTTPD.jks

Location: /usr/WebSphere/AppServer/etc (or the directory of your choice)

4. At the Password Prompt window, enter the password of your choice. The more random the password, the higher the password strength. There is no requirement to stash the password to a file with the JKS capable ikeyman version.
5. As we are only going to be implementing a peer-to-peer SSL connection between the embedded HTTP server and the Web server plug-in, we are not concerned with the signer certificates of the publicly circulating root Certificate Authorities (CA). As such, optionally delete all of the CA trusted signer certificates.
6. From the ikeyman menu bar, select **Create -> New Self-Signed Certificate** to create a new self-signed certificate key pair. The following options then need to be specified; you may choose to complete all of the remaining fields for the sake of completeness:

Key Label: EmbeddedHTTPD

Version: X509 V3

Key Size: 1024

Common Name: rs617002.itso.ra1.ibm.com

Organization: IBM

Country: US

Validity Period: 365

Click **OK** when you are finished.

7. Extract the public self-signed certificate key, as this will be used later by the Web server plug-in peer to authenticate connections originating from the embedded HTTPD resident in the managed Application Server Web Container.
8. Select **Personal Certificates** in the ikeyman drop-down menu and select the **EmbeddedHTTPD** certificate that was just created.

9. Click the **Extract Certificate** button, ensuring **EmbeddedHTTPD** remains selected. Extract the certificate to a file:
Data type: Base64-encoded ASCII data
Certificate file name: EmbeddedHTTPDPubCert.arm
Location: /usr/WebSphere/AppServer/etc (or the directory of your choice)
Click **OK** when you are finished.
10. Close the key database and quit ikeyman when you are finished

11.4.3 Exchanging public certificates

The following two sections will show you how to exchange certificates between the Web Container keystore and the Web server plug-in keyfile.

Exchanging the Web Container public certificate with the Web server plug-in keyfile

1. Back on the Web server machine, launch the IBM ikeyman tool that ships as part of GSKit and supports the CMS key database format.
2. From the ikeyman menu bar, select **Key Database File -> Open** and select the previously created key database file **WASV4Plugin.kdb**.
3. At the Password Prompt window, enter the password and click **OK** when you are done.
4. Select **Signer Certificates** in the drop-down menu and click the **Add** button. This will allow you to import the public certificate previously extracted from the embedded HTTP server/Web Container keystore. Adding the embedded HTTP server public key certificate as a trusted signer will enable authentication of connections originating from the embedded HTTP server (Web Container) peer.
Data type: Base64-encoded ASCII data
Certificate file name: EmbeddedHTTPDPubCert.arm
Location: /usr/WebSphere/AppServer/etc (or the directory of your choice)
Click **OK** when you are finished.
5. You will then be prompted for a label name by which the trusted signer public certificate will be known. Enter a label for the certificate:
DefaultAppWebContainer.
6. Close the key database and quit ikeyman when you are finished.

Exchanging the public certificate from the Web server plug-in keyfile with the Web Container keystore

1. Launch the IBM JKS capable ikeyman version that ships under the WebSphere bin directory
2. From the ikeyman menu bar, select **Key Database File -> Open** and select the previously created **EmbeddedHTTPD.jks** keystore file.
3. At the Password Prompt window, enter the password and click **OK** when you are done.
4. Select **Signer Certificates** in the drop-down menu and click the **Add** button. This will allow you to import the public certificate previously extracted from the Web server plug-in certificate database/keyfile.

Data type: Base64-encoded ASCII data

Certificate file name: WASV4PluginPubCert.arm

Location: /usr/WebSphere/AppServer/etc (or the directory of your choice)

Click **OK** when you are finished.

5. You will then be prompted for a label name by which the trusted signer public certificate will be known. Enter a label for the certificate: RemotePlugin.
6. Close the key database and quit ikeyman when you are finished.

11.4.4 Modifying the Web server plug-in configuration file

After creating the certificate keys for authenticating SSL between the Web Server plug-in and the peer Web Container, the plug-in configuration file must be modified to reference the plug-in keyring and the password stash file. This allows the transport protocol to be changed from HTTP to HTTPS, using the certificates stored in the keyring.

In this case, a standard non-secure HTTP connection, as depicted below:

```
<Transport Hostname="rs617001" Port="9080" Protocol="http"/>
```

changes to:

```
<Transport Hostname="rs617001" Port="9080" Protocol="https">  
  <Property name="keyring" value="/usr/WebSphere/AppServer/etc/Plugin.kdb"/>  
  <Property name="stashfile"  
value="/usr/WebSphere/AppServer/etc/Plugin.sth"/>  
</Transport>
```

One option that you may consider implementing is the segregation of HTTP and HTTPS requests originating from a client Web browser, as it makes little sense to encrypt traffic between the Web server plug-in and a Web Container, if the initial connection to the Web server remains unsecured.

In this case, you can create two different virtual hosts within WebSphere, one capable of handling non-secured HTTP requests, typically on port 80, and the other dedicated to handling HTTPS traffic defaulting to port 443. Then, when you edit the plug-in configuration file, you can specify HTTPS for the virtual host associated with Web Server HTTPS traffic.

For administrators using the IBM HTTP Server, the location of the plug-in configuration file `plugin-cfg.xml` is specified by the `WebSpherePluginConfig` directive in `httpd.conf`.

The complete `plugin-cfg.xml` as used in our example is shown in Example 11-19.

Example 11-19 Sample plugin-cfg.xml

```
<?xml version="1.0"?>
<Config>
  <Log LogLevel="Error" Name="/usr/WebSphere/AppServer/logs/native.log"/>
  <VirtualHostGroup Name="default_host">
    <VirtualHost Name="rs617001.itso.ral.ibm.com:80"/>
    <VirtualHost Name="rs617001.itso.ral.ibm.com:9080"/>
    <VirtualHost Name="rs617001.itso.ral.ibm.com:443"/>
  </VirtualHostGroup>
  <ServerGroup Name="rs617001/Default Server">
    <Server CloneID="t7hibiq9" Name="Default Server">
      <Transport Hostname="rs617001" Port="9080" Protocol="https">
        <Property name="keyring"
value="/usr/WebSphere/AppServer/etc/ITS02/PluginKeyring.kdb"/>
        <Property name="stashfile"
value="/usr/WebSphere/AppServer/etc/ITS02/PluginKeyring.sth"/>
      </Transport>
    </Server>
  </ServerGroup>
  <UriGroup Name="rs617001_sampleApp/default_app_URIs">
    <Uri Name="/servlet/snoop/*"/>
    <Uri Name="/servlet/snoop"/>
    <Uri Name="/servlet/snoop2/*"/>
    <Uri Name="/servlet/snoop2"/>
    <Uri Name="/servlet/hello"/>
    <Uri Name="/ErrorReporter"/>
    <Uri Name="*.jsp"/>
    <Uri Name="*.jsv"/>
    <Uri Name="*.jsw"/>
    <Uri Name="/j_security_check"/>
    <Uri Name="/servlet/*"/>
  </UriGroup>
</Config>
```

```

</UriGroup>
<UriGroup Name="rs617001_sampleApp/examples_URIs">
  <Uri Name="/webapp/examples"/>
</UriGroup>
<Route ServerGroup="rs617001/Default Server"
  UriGroup="rs617001_sampleApp/default_app_URIs"
VirtualHostGroup="default_host"/>
  <Route ServerGroup="rs617001/Default Server"
  UriGroup="rs617001_sampleApp/examples_URIs"
VirtualHostGroup="default_host"/>
</Config>

```

11.4.5 Modifying the Web Container to support SSL

To complete the SSL configuration between Web server plug-in and Web Container, the WebSphere Web Container must be modified to use the previously created self-signed certificates. For this reason, this section assumes you have created the certificate keys for SSL authentication. If you have not done so, go back now and complete the certificate generation and key exchange tasks.

The following steps document the required Web Container modifications.

1. Start the WebSphere Administrative console and select the managed **Application Server** from the WebSphere topology tree containing the Web Container that you wish to secure with SSL.
2. With the managed Application Server of your choice selected, hold down the right mouse key and click the **Properties** button. This will launch a new window containing the Application Server properties, as shown in Figure 11-32.

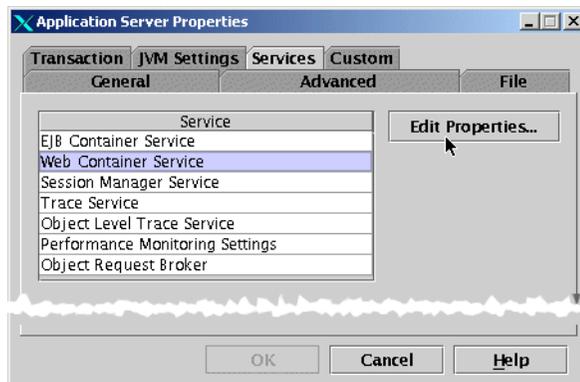


Figure 11-32 Application Server Properties window

3. Select the **Services** tab and then the **Web Container Service**, as shown above in Figure 11-32. Clicking **Edit Properties** will launch the Web Container Service window.

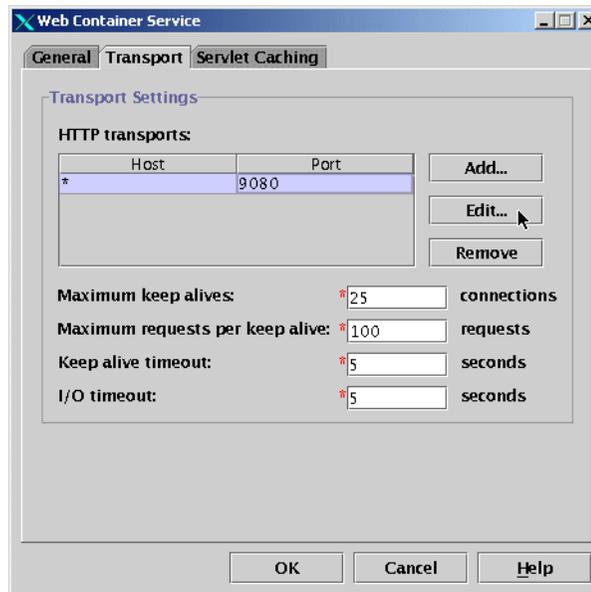


Figure 11-33 Web Container Service window

4. Detailed under the Transport tab is the HTTP transport mechanism unique to this Web Container. The Host field denotes the source from which requests can originate and the Port the listening socket to which requests are sent from the Web server plug-in.

It is possible to configure more than one port per Web Container. However, if two or more ports are set to the same protocol, such as HTTP. Only the first port will be used by the plug-in in the Web server for servicing requests. What this allows, in effect, is for one port to be configured for non-secured HTTP while a second port can service HTTPS requests.

5. Select the HTTP Transport Properties specific to TCP port 9080 and click **Edit**.

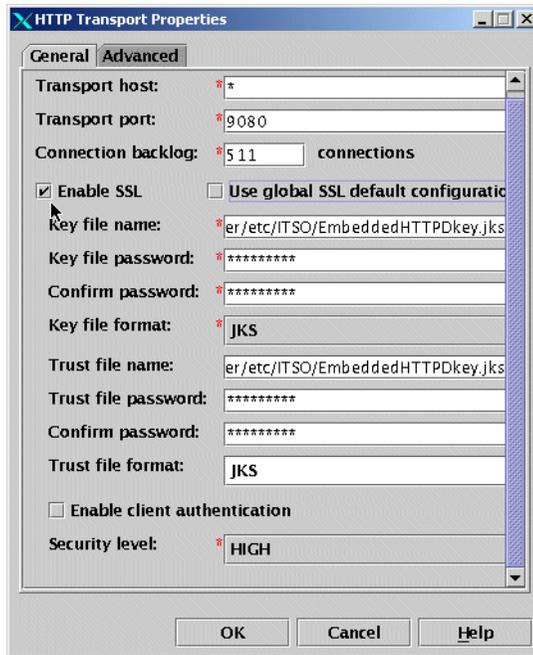


Figure 11-34 HTTP Transport Properties window

6. In the HTTP Transport Properties window, specify the full path and file name to the certificate keyring previously created for the Web Container in the Key file name, as such: `/usr/WebSphere/AppServer/etc/EmbeddedHTTPD.jks`.

Enter the Key file password as previously set with `ikeyman` for the keyfile.

Ensure the Key file format remains set to support the Java keystore format: JKS.

Configure the Trust file name, Trust file password and Trust file format to match the values inserted for the Key fields. The concept of differing files for Key and Trust association is explained later in this chapter.

Enabling client authentication is optional. If enabled, the Web Container will request that the Web server plug-in authenticate itself. The concept can best be demonstrated by pointing a Web browser directly at the embedded HTTP server, bypassing the Web server entirely. Here, as shown in Figure 11-35, the Web Container will prompt the client for a certificate for authentication. Previously, we cross-exchanged the public keys of the Web server plug-in and Web Container, so both server and client authentication are valid.



Figure 11-35 Internet Explorer Client Authentication window

The Security level should remain set to HIGH to ensure that SSL uses 128 bit encryption.

Click **OK** when you are finished.

7. After the HTTP Transport Properties window closes, click **OK** in the Web Container window. Likewise, click **OK** in the Application Server Properties window. Finally, click **Apply** under Application Server to save the changes.
8. After stopping and starting both the Web server and the WebSphere Application Server, SSL will encrypt the previously non-secured connection.

11.5 Restricting access to only HTTPS connections

In the event that you wish to restrict connections to only the secure HTTPS protocol, you can implement a *transport guarantee* as a security constraint on the resource in question. As such, the following steps must be taken:

1. Launch the WebSphere Application Assembly Tool (AAT).
2. This step can be undertaken either before an enterprise application archive .ear file has been deployed into WebSphere or after deployment into WebSphere. If the latter option is your choice, open the expanded archive correlating to the enterprise application archive in question, found under the WebSphere installedApps directory. This option is discouraged in a production environment.
3. Expand both the Web Modules icon and the subsequent application that contains the resources on which you wish to implement the security constraint.

4. Finally, by further expanding the Security Constraints icon, you can select each permissible resource and specify the appropriate constraint as required. Figure 11-36 illustrates this for the show config servlet ShowCfg.



Figure 11-36 Sample security constraint

5. Select the **Transport guarantee** as required. This dictates the level at which communication between a client and WebSphere must be established for a resource to be served. The options are as follows:
 - None means that the application does not require any transport guarantee.
 - Integral means that the application requires that the data sent between the client and the server be sent in such a way that it cannot be changed in transit.
 - Confidential means that the application requires that the data be transmitted in a way that prevents other entities from observing the contents of the transmission.
6. Click **OK** and then save the changes made with Application Assembly Tool (AAT).
7. If the modification was made to a resource already deployed into WebSphere, stop and start the associated Application Server containing resource for the security constraint modifications to be included into the runtime.

If you now try and access a resource upholding the HTTPS security constraint not using the HTTP protocol, you will receive the following warning:

HTTP Error 403 - Forbidden, You are not authorized to view this page error.

Certain versions of Microsoft Internet Explorer will show the response, as shown in Figure 11-37.

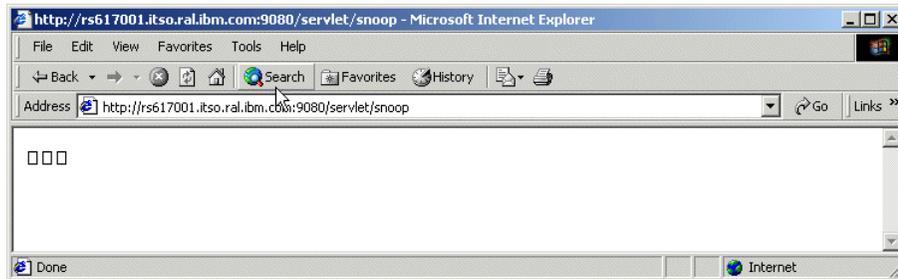


Figure 11-37 False response

Note: You may consider placing an `ErrorDocument 403` directive in your `httpd.conf` file, to present to users with an alternative customized error message/HTML page.

11.6 Securing WebSphere LTPA with SSL

SSL can be used to secure communication between WebSphere and your chosen LDAP Directory Server, thus protecting LTPA user authentication requests. By default, this connection is insecure, even when WebSphere Global Security is enabled.

In the following section, we will demonstrate the tasks necessary to secure WebSphere against three different LDAP Directory Server solutions:

- ▶ IBM SecureWay Directory Server (on AIX)
- ▶ iPlanet Directory Server (installed on Sun Solaris)
- ▶ Lotus Domino (installed on Windows 2000)

11.6.1 IBM SecureWay Directory Server

Administrators not familiar with Global Security should recognize that WebSphere offers two options for managing the SSL certificates associated with securing the WebSphere-to-LDAP connection.

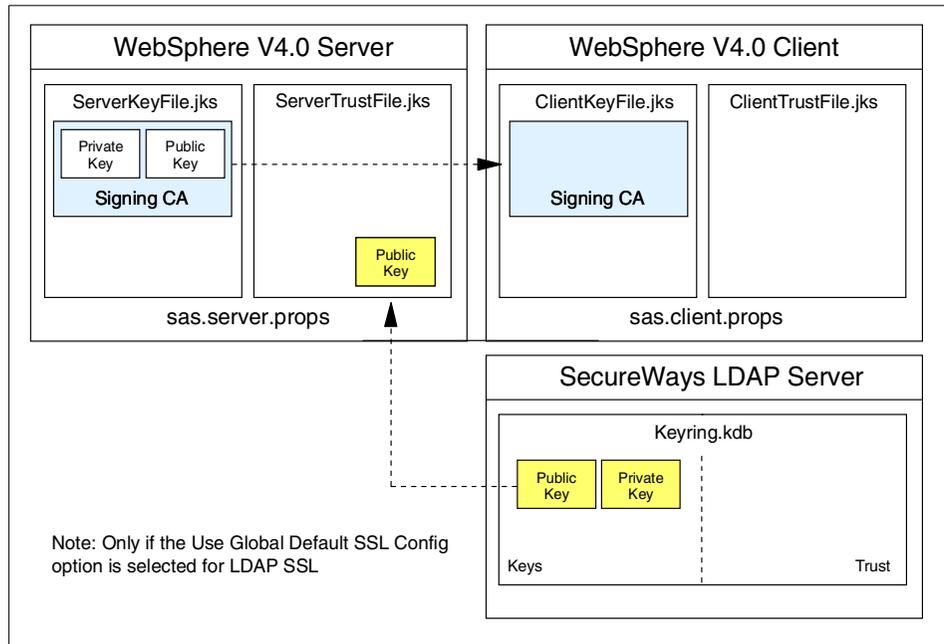


Figure 11-38 Option 1: using the default SSL Key and Trust Files

The first, as depicted in Figure 11-38, highlights the scenario whereby the public certificate key associated with the remote LDAP Directory server is effectively placed into the same certificate key database as used by the WebSphere internal security mechanism. Here, the WebSphere certificate key database actually constitutes two Java Key Store (JKS) compliant files; the ServerKeyFile.jks and the ServerTrustFile.jks, respectively. Both files and their contents were previously discussed in Section 11.1.1, “The Demo Keyring” on page 217.

If you wish to centralize all of your WebSphere SSL certificates into a single key database, then you can select this option by using the Default SSL settings from the LDAP SSL Configuration window. You will recall that the same is also true when configuring the SSL settings associated with a WebSphere Web Container. That is, you can use the Default SSL settings/key database to store your Web Container and Web server plug-in certificates.

Alternatively, the second option, as shown Figure 11-39, is to create a separate SSL key database solely for the purpose of containing the LDAP Directory public certificate key. This method allows you to isolate the key database to the particular function of securing the WebSphere-to-LDAP connection. You should select this option as your initial starting point, if you are not familiar with the intricacies of WebSphere Global Security.

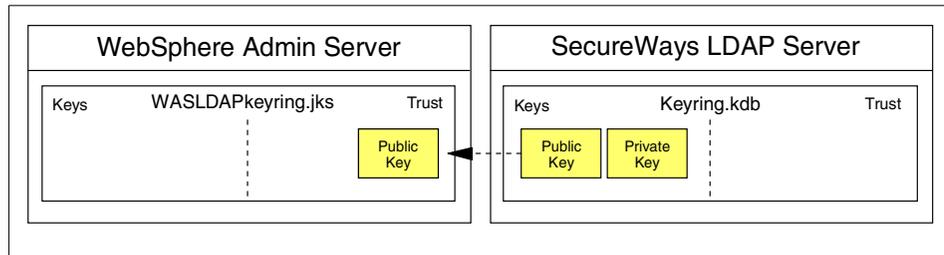


Figure 11-39 Option 2: using dedicated LDAP key and files

Note that in Figure 11-39 the key database is a single Java Key Store (JKS) file named WASLDAPKeyring.jks. This is acceptable, because while the Java Secure Sockets Extension (JSSE) standard permits the use of separate files for certificate keys and certificate trusted keys, it also allows you to amalgamate both entities into a single file. The words *Keys* and *Trust* are shown to document this distinction.

We adopted the second approach, as shown in Figure 11-39, in the following worked example.

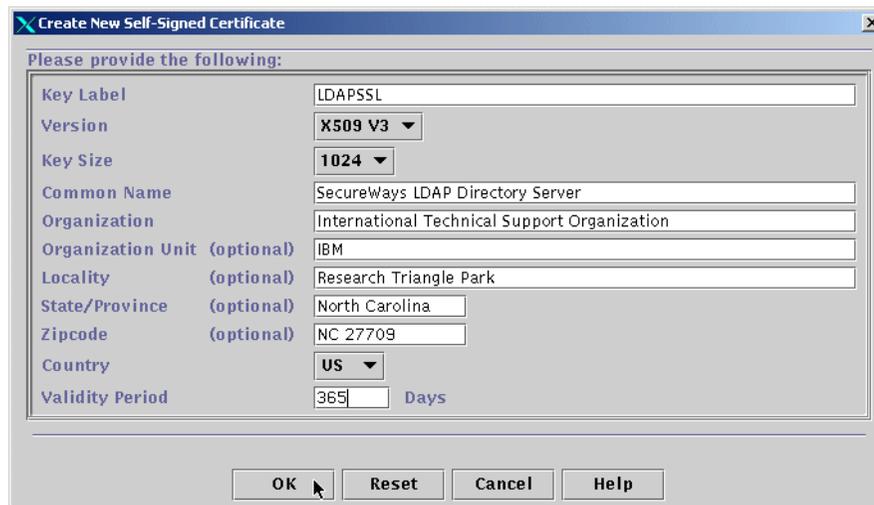
11.6.2 Creating a self-signed certificate for the SecureWay LDAP peer

The instructions that follow in this section are identical to the tasks documented in Section 11.4.1, “Generating a self-signed certificate for the Web server plug-in” on page 271. The steps are nevertheless repeated here for the sake of completeness.

1. Create a suitable directory on the SecureWay LDAP Directory server for storing the keyfile that will contain the SSL certificate.
2. Launch the IBM ikeyman tool that ships as part of IBM GSKit and supports the CMS key database format. This is not the JSSE ikeyman version that ships in the WebSphere bin directory.
3. Create a new key database to store the certificate. You can manage multiple certificates in a single key database. From the ikeyman menu bar, select **Key Database File -> New**.
4. Apply the following settings and click **OK** when you are done.
 - Key database file: CMS key database file
 - File name: SecureWayLDAP.kdb
 - Location: /usr/ldap/conf
5. At the Password Prompt window, enter the password of your choice. The more random the password, the higher the password strength. Stashing the

password to a file is optional with the SecureWay LDAP Directory server, as you can specify the password in the SecureWay LDAP SSL configuration settings. Click **OK** when you are done.

6. The assumption is made that because we are only implementing a peer-to-peer SSL connection between the SecureWay LDAP Directory server and WebSphere, we are not concerned with the signer certificates of the publicly circulating root Certificate Authorities (CAs). In this case, we can optionally delete all of the CA trusted root certificates listed under the Signed Certificates menu in ikeyman.
7. From the ikeyman menu bar, select **Create -> New Self-Signed Certificate** to create a new private/public self-signed certificate key pair. The following options, as shown in Figure 11-40, then need to be specified. Supplement your own values accordingly. You do not need to ensure that the certificate Common Name equals the fully qualified host name, as the certificate will not be signed by a third-party Certificate Authority (CA). Nevertheless, this still remains good practice.



The screenshot shows a dialog box titled "Create New Self-Signed Certificate". It contains the following fields and values:

Field	Value
Key Label	LDAPSSL
Version	X509 V3
Key Size	1024
Common Name	SecureWays LDAP Directory Server
Organization	International Technical Support Organization
Organization Unit (optional)	IBM
Locality (optional)	Research Triangle Park
State/Province (optional)	North Carolina
Zipcode (optional)	NC 27709
Country	US
Validity Period	365 Days

Buttons at the bottom: OK, Reset, Cancel, Help.

Figure 11-40 Creating a new self-signed certificate

Click **OK** when you are finished.

8. Extract the public self-signed certificate key, using **Extract certificate**, as this will be used later by WebSphere to encrypt LDAP authentication requests sent to the SecureWay LDAP Directory server. A good analogy to the public key is an open safe. When WebSphere closes the safe, only the private key resident at the LDAP server can unlock the safe and retrieve the data.

9. Select **Personal Certificates** in the ikeyman drop-down menu and select the **LDAPSSL** certificate. Click the **Extract Certificate** button, ensuring that **LDAPSSL** remains selected. In the window entitled Extract a Certificate to a File, enter the values as shown below and click **OK** when you are finished. Supplement your own values where necessary.

Data type: Base64-encoded ASCII data

Certificate file name: SecureWayLDAPPubCert.arm

Location: /usr/ldap/conf

10. Close the key database and quit ikeyman when you are finished.

11.6.3 Creating a key database for the WebSphere LDAP SSL peer

The following steps will show how to create a key database for the WebSphere LDAP SSL peer.

1. Launch the IBM JSSE ikeyman utility that ships under the WebSphere *bin* directory. On Unix systems, this is invoked by running the **ikeyman.sh** script.
2. Create a new database to store the public certificate from the remote LDAP server. From the ikeyman menu bar select **Key Database File -> New**.
3. In the window that appears, ensure that the key database file is set to support the Java Key Store (JKS) format. Complete the file name and location fields appropriately. Then click **OK** when you are done.

Key database file: JKS

File name: WASLDAPKeyring.jks

Location: /usr/WebSphere/AppServer/etc

4. At the Password Prompt window, enter the password of your choice. The more random the password, the higher the password strength. There is no requirement to stash the password to a file with the JSSE ikeyman utility.
5. No requirement exists to authenticate against any of the default trusted root Certificate Authority (CA) certificates installed under the ikeyman Signer Certificates menu. In this case, we can optionally delete all of the CA trusted root certificates.
6. Remaining in the Signer Certificates menu, click the **Add** button. This will allow you to import the public certificate previously extracted from the remote SecureWay LDAP certificate database (SecureWayLDAP.kdb). The assumption is made that you have successfully transferred the public certificate key (SecureWayLDAPPubCert.arm) from the remote SecureWay LDAP Directory Server. Complete the fields and click **OK** when you are done.

Data type: Base64-encoded ASCII data

Certificate file name: SecureWayLDAPPubCert.arm

Location: /usr/WebSphere/AppServer/etc

7. The act of introducing a new certificate into the key database will prompt you to specify a label name or alias for the entry; type in secureway. The label name will be listed under the Signer Certificate menu in ikeyman.

Note: We suggest that you refrain from using spaces in the label name.

8. After checking that the certificate successfully installed as a trusted signer, close the key database and quit ikeyman.

11.6.4 Modifying the SecureWay LDAP Directory to use SSL

After successfully generating and exchanging the SecureWay public key, both the SecureWay Directory Server and WebSphere must be configured to support SSL. The assumption is made that you have previously installed and configured the SecureWay LDAP Directory for authenticating WebSphere users, albeit without SSL.

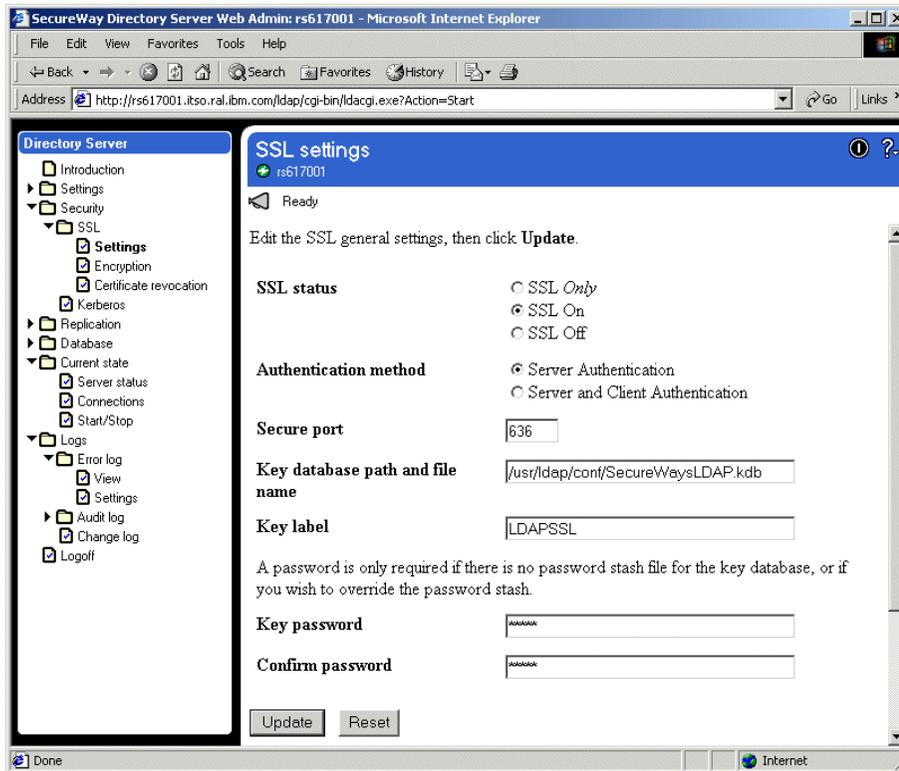


Figure 11-41 Configuring SSL for the SecureWay LDAP Directory Server

Complete the following steps to enable LDAP SSL communication.

1. Launch the SecureWay Web-based administration console in your chosen Web browser. Open the URL `http://<servername>/ldap`.
2. Complete the LDAP Administrator ID and Password fields when prompted to authenticate yourself to the Directory Server. Typically, the Distinguished Name (DN) `cn=root` is used here. However, any user with sufficient privileges can perform this task.
3. From the Directory Server topology tree in the left pane, select the **SSL Settings** tab found under the Security heading. Example 11-41 shows the corresponding SSL Settings window that will be displayed in the right-hand-side pane.

4. You must complete the following fields to enable LDAP communication over SSL:
 - i. **SSL status:** select **SSL On** or **SSL Only** if you wish to prevent non-SSL LDAP connections.
 - ii. **Authentication method:** select **Server Authentication**. You may choose to select **Server and Client Authentication**, in which case you need to ensure that the public certificate key associated with any authenticating client is resident in your (LDAP) certificate key database.
 - iii. **Secure port:** select **636** which is the Internet standard for secure LDAP communication. You may choose any port that is not in use. On Unix, only the root has access to the ports below 1024 by default.
 - iv. **Key database path and file name:** specify the fully qualified file name of the CMS key database previously created. In our example, this is set to `/usr/ldap/conf/SecureWayLDAP.kdb`. SecureWay does not support the Java Key Store (JKS) type certificate key database.
 - v. **Key label:** as a key database can contain multiple certificates, specify the label name of the certificate used for authenticating the LDAP Directory Server. In our example this is set to `LDAPSSL`.
 - vi. **Key password:** specify the key database password if you did not generate a password stash file when creating the certificate key database originally. This password will be used by SecureWay to gain access to the certificate database.
5. Click the **Update** button when you have completed all of the above fields.
6. For the changes to be included into the runtime, you must stop and restart the LDAP Directory Server. Once restarted, you can check the status of the Directory by expanding the **Current State** and **Server Status** menus. If the Directory fails to start, check the Error logs. Unix users can also check that the Directory is listening for incoming SSL LDAP connections by using the `netstat -a` command and grepping for port 636.

If you are concerned with the level of SSL support offered by the SecureWay LDAP Directory Server, you can choose to restrict the permitted encryption algorithms. For example, you may decide that (40-bit) encryption is inadequate for your SSL implementation. In this case, the (40-bit) encryption method can be deselected, as shown below in Figure 11-42.

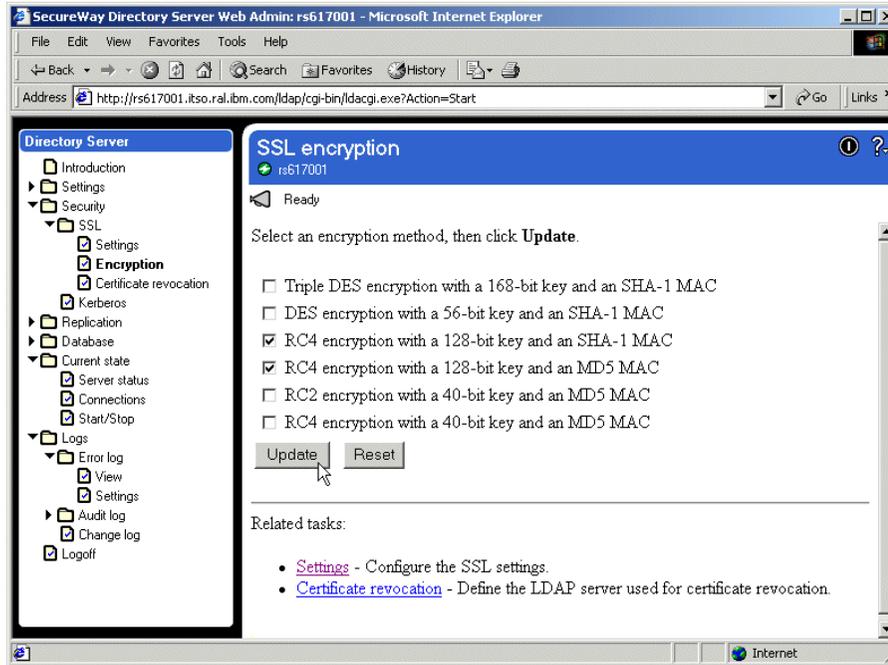


Figure 11-42 Configuring SSL encryption for LDAP

Bear in mind that each SSL peer must support the same encryption method/cipher suite to be able to establish an encrypted session. The encryption methods supported by WebSphere are classified into three categories *high*, *medium* and *low*, and are configured via the Security level setting on the LDAP SSL Configuration window as shown in Figure 11-44 on page 293.

11.6.5 Modifying WebSphere to use LDAP over SSL

Here, the assumption is again made that you have previously configured WebSphere to successfully authenticate users against the SecureWay LDAP Directory Server without using SSL for securing the WebSphere-to-LDAP connection. The LDAP Distinguished Name (DN) and LDAP topology structure do not need to be modified in any way to support SSL.

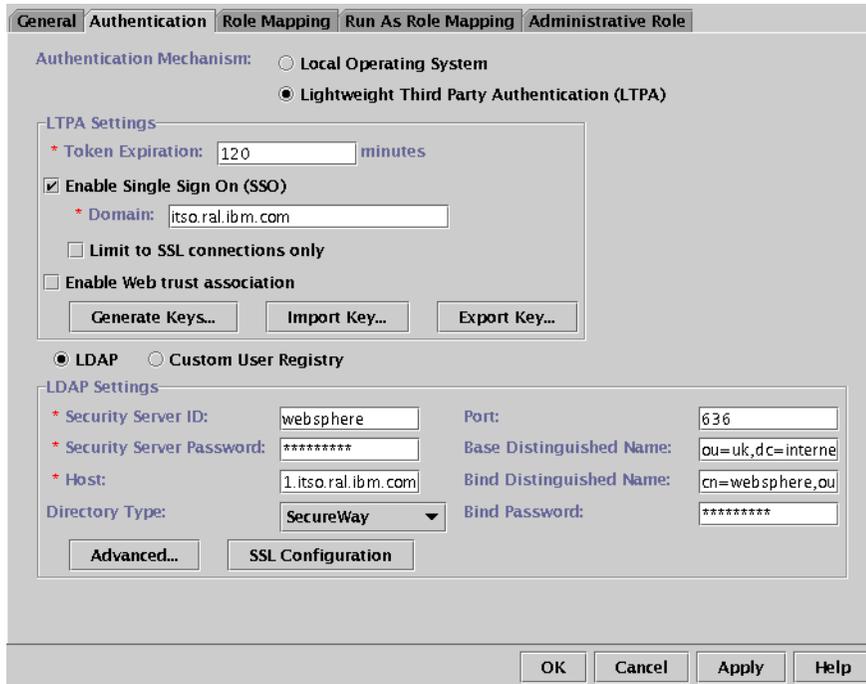


Figure 11-43 Security Center

Note: Care should be taken when specifying the LDAP settings, as the Security Server ID and Base/Bind Distinguished Names are often misinterpreted.

Follow these steps to configure WebSphere to support LDAPS (LDAP over SSL):

1. Start the WebSphere administrative console and launch the Security Center. Select the **Authentication** tab to view the Lightweight Third Party Authentication (LTPA) and the LDAP settings, as shown above in Figure 11-43.

As a prerequisite, you should only proceed to configure SSL if you have previously enabled non-secure LDAP authentication. This recommendation is made to isolate problem determination to the SSL transport mechanism, if the configuration should fail.

2. Apart from the SSL configuration parameters, which we will cover in the next step, and the LDAP port number, all of the settings should remain unmodified as per the non-secure LDAP authentication scheme. The settings and expected values are discussed in detail below:

- i. **Security Server ID:** specify the user you have created in your LDAP implementation that will represent the WebSphere Server identity; we have used the websphere ID. Do not use the LDAP cn=root entity under any circumstances.
 - ii. **Security Server Password:** specify the corresponding password for the Security Server user, in our example: websphere.
 - iii. **Host:** this is the fully qualified host name of the LDAP Directory server, rs617001.itso.ral.ibm.com.
 - iv. **Directory Type:** select **SecureWay** on this occasion. Note that if you modify any of the filtering rules, the Directory Type will change to **Custom** even though SecureWay remains the LDAP Directory of your choice.
 - v. **Port:** specify **636** which corresponds to the TCP/IP port listening for SSL enabled LDAP queries on the remote SecureWay LDAP Directory.
 - vi. **Base Distinguished Name:** set this to the entry point into your LDAP Directory Server where WebSphere will search for authenticated users. We have used: ou=uk,dc=internetchaos,dc=com.
 - vii. **Bind Distinguished Name:** specify the fully qualified Distinguished Name of the LDAP user that has sufficient privileges to search and authenticate WebSphere users in the LDAP Directory Server. If you have restricted anonymous searches on the WebSphere user name space in your LDAP Directory, this field must be completed regardless of whether or not the user is the same as that specified for the Security Server ID. Our Bind Distinguished Name was:
cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com.
 - viii. **Bind Distinguished Password:** specify the corresponding password for the Bind Distinguished user, in our example: websphere.
3. At this point *do not* click **Apply** or **OK**.
 4. Proceed to the LDAP SSL Configuration menu by selecting the **SSL Configuration** button. Figure 11-44 shows the resulting window that you will be presented with.

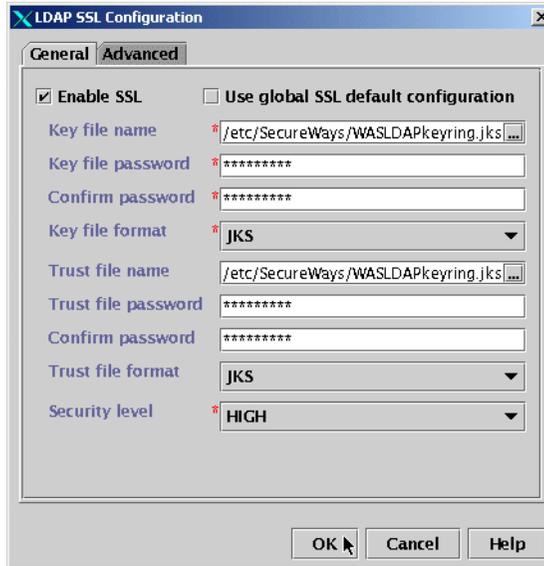


Figure 11-44 LDAP SSL configuration

5. Select the **Enable SSL** box and complete the fields that follow to reference your newly created Java Key Store (JKS) compliant certificate key database.
 - i. **Key file name:** specify the fully qualified file name of the Java Key Store (JKS) key database previously created. In our example this is set to `/usr/WebSphere/AppServer/etc/WASLDAPKeyring.jks`.
 - ii. **Key file password** and **Confirm password:** state the password used to protect the Java Key Store (JKS) certificate key database above.
 - iii. **Key file format:** ensure that **JKS** is selected.
 - iv. **Trust file name:** potentially, you can set this to point at a second Java Key Store (JKS) used for holding trusted certificate keys. However, if you choose not to differentiate between personal keys and trusted keys, and opt to use a single key database for both tasks, this field should be set to the same value as the Key file name. In our example, this is set to `/usr/WebSphere/AppServer/etc/WASLDAPKeyring.jks`.
 - v. **Trust file password** and **Confirm password:** state the password used to protect the Java Key Store (JKS) certificate key database above.
 - vi. **Trust file format:** ensure that **JKS** is selected.

- vii. **Security level:** setting this to **High** will ensure that the strongest SSL encryption algorithms are used for secure communication. The setting must be compatible with algorithms supported by the SSL peer.
 - viii. Click **OK** when you are done.
6. You will now be returned to the main WebSphere Security Center Authentication window. Double-check the LDAP settings and click **OK**.
 7. If your configuration proves successful, SSL communication will be established between WebSphere and the SecureWay LDAP Directory server at this point. In this case, you can click **Apply** in the WebSphere Security Center window and then proceed to restart WebSphere so the changes can be included in the next runtime.

If, upon clicking **OK**, the configuration fails with an error message and exception, double-check all parameters and certify that the remote LDAP Server is indeed running. Do not click **Apply** or terminate the WebSphere Admin Client console at this point if you experience an error, as you will lose the ability to undo the changes that you made.

11.6.6 Disabling SecureWay Anonymous LDAP searches

In a production environment, it is envisaged that you may wish to prevent anonymous LDAP searches of the WebSphere user space, although such searches typically only reveal non-sensitive information about a user. The very fact that any user information can be retrieved at all may pose a security risk.

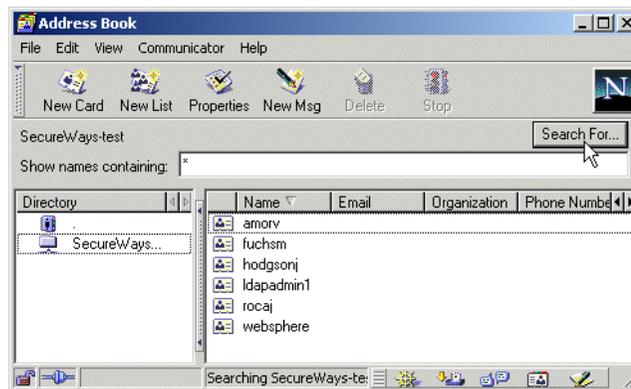


Figure 11-45 Netscape - Address book

The Netscape Communicator 4.7 Address Book can be used to demonstrate this argument. Figure 11-45 shows that with a little knowledge about the remote LDAP server, it is possible to retrieve the WebSphere authentication user registry. Note that detailed in the listing are also the administrative users *websphere* and *ldapadmin1*.

The LDAP Directory schema

Throughout this chapter, we have used a fictitious LDAP Distinguished Name (DN) to demonstrate the tasks involved with configuring WebSphere V4.0 Global Security LTPA authentication. Furthermore, we have deliberately avoided using the *o=organization, c=country* suffix as documented in many a publication, in preference for a domain name syntax-based DN. As such, our complete LDAP Directory hierarchy matches that shown in Figure 11-46.

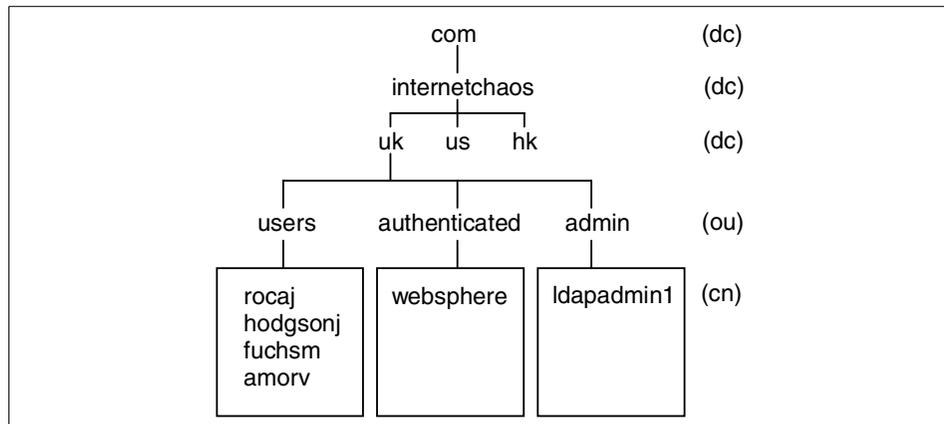


Figure 11-46 LDAP Directory schema

We anticipated that three special organizational units (OU) would be needed in our configuration. The first would be used to simply hold the WebSphere authenticating users and as such correlate to the Distinguished Name (DN): `ou=users,ou=uk,dc=internetchaos,dc=com`.

A second organizational unit (OU) was created to hold the WebSphere Security Server ID. We also granted the identity the right to search and authenticate entities present in the users' organization unit, but restricted any modification rights. As such the Distinguished Name (DN) matches: `ou=authenticated,ou=uk,dc=internetchaos,dc=com`.

Finally, it was envisaged that a number of users would be responsible for maintaining the actual underlying LDAP Directory, adding and performing administration tasks on entities in the users' organizational unit (OU). This third organization unit (OU) assumed the Distinguished Name (DN):
ou=admin,ou=uk,dc=internetchaos,dc=com.

The instructions for configuring the precise Access Control Lists (ACLs) relating to the above scheme follow.

Restricting access to the users organizational unit (OU)

The following steps will guide you through the process of restricting access to the users' organizational unit (OU).

1. Launch the SecureWay Directory Management Tool (DMT) and, if necessary, rebind as an authenticated user, with adequate administration privileges.
2. Select the **Browse Tree** option found under the Directory tree expandable menu in the management topology tree. The resulting window will look similar to that shown in Figure 11-47.

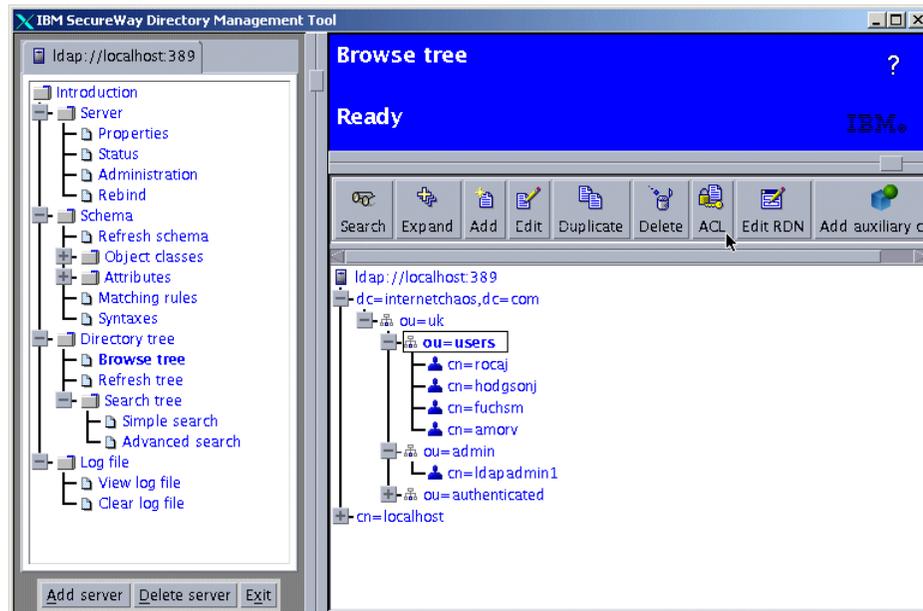


Figure 11-47 Browsing the directory

3. Select the relative distinguished name (RDN) representing the users' organizational unit (ou=users) and click the **ACL** button on the DMT menu bar. Select the **ou=users** entry under **ou=uk,dc=internetchaos,dc=com**.

4. In the resulting window, ensure that the DN entry access control list (ACL) source corresponds to the organizational unit (OU) specified in step 3 and select the **Descendant directory tree entries inherit from this entry** box. This will cascade the ACL modifications to the entities (users) that lie beneath this organizational unit (OU).
5. By default, the Subject Distinguished name (DN) will be set to *CN=ANYBODY*; this is equivalent to anonymous access. Select the field where CN=ANYBODY appears and modify the entry to read: *cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com*.
Change the Type to **access-id** and click **Add**, as shown below in Figure 11-48.

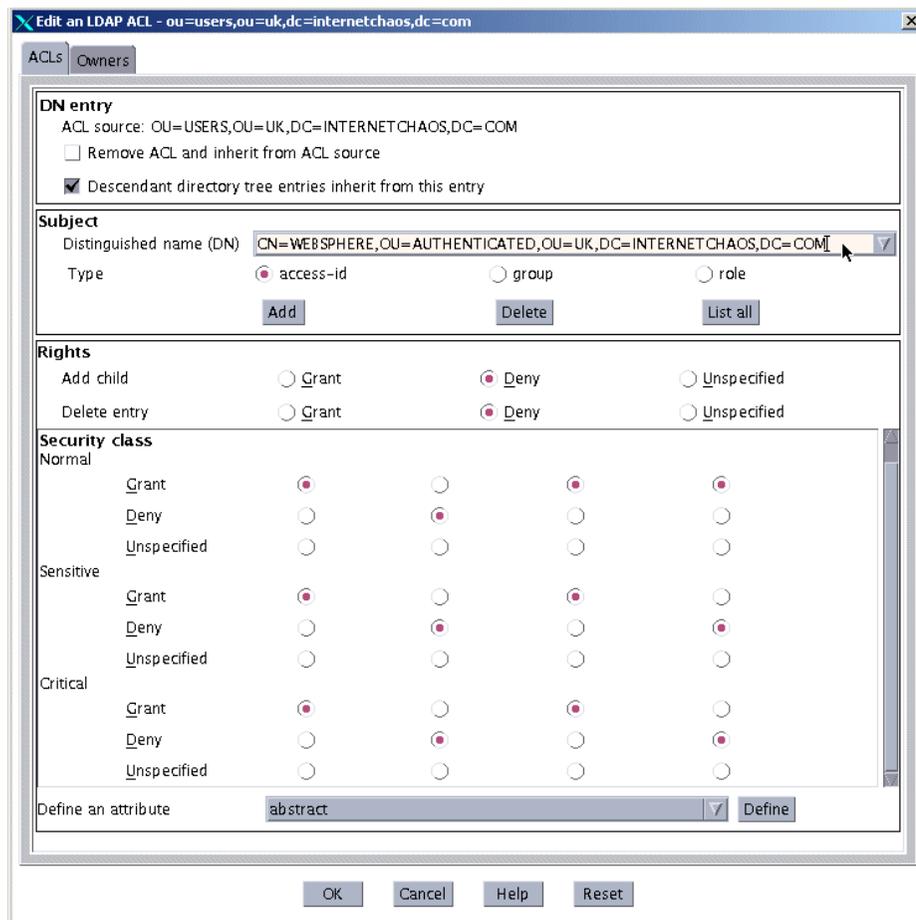


Figure 11-48 LDAP ACL settings

6. The new entry will now appear in the Subject drop-down Distinguished Name menu. Select **CN=ANYBODY** once more, and click **Delete**. This will remove the entry that by default is currently allowing anonymous connections to view the entities (WebSphere users) under the *users* ' organizational unit (OU).
7. We want to give just enough privileges to the websphere administrative user *cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com* to perform authentication queries on the entities (WebSphere users) found under *ou=users,ou=uk,dc=internetchaos,dc=com*.

In the **Rights** section, deny the websphere administrative user the privilege to **Add child** and **Delete an entry**. For the security class, only **grant Read** and **Search** to *cn=websphere*, deny all others access. Do this for **Normal**, **Sensitive** and **Critical** data types.
8. Click **OK** when you are done.

Restricting access to the authenticated organizational unit (OU)

When we initially created the LDAP Directory, a separate organizational unit was created to contain the user (*cn=websphere*). This user serves two purposes in our implementation. First, it fulfills the requirement for the WebSphere Security Server ID, the effective identity which WebSphere assumes when running. Secondly, it performs the LDAP authentication searches by being specified as the LDAP Bind Distinguished Name. Isolating the entity from the regular users' organization unit allows different access rights and privileges to be defined.

The steps that follow detail the tasks necessary to prevent anonymous access to the authenticated organizational unit and by virtue that *cn=websphere* lies beneath *ou=authenticated*, to enable LDAP access for the WebSphere administrative user.

1. Select the Relative Distinguished Name (RDN) representing the *ou=authenticated* organizational unit and click the **ACL** button on the DMT menu bar. Select the **ou=authenticated** entry under *ou=uk,dc=internetchaos,dc=com*.
2. In the resulting window, ensure that the DN entry ACL source corresponds to the organizational unit (OU) specified above and select the **Descendant directory tree entries inherit from this entry** box. This will cascade the ACL modifications to the entities (users) that lie beneath this organizational unit (OU).
3. By default, the Subject Distinguished name (DN) will be set to **CN=ANYBODY**; this is equivalent to anonymous access. Select the field where **CN=ANYBODY** appears and modify the entry to read: *cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com*.

Change the Type to **access-id** and click **Add**, as shown below in Figure 11-49.

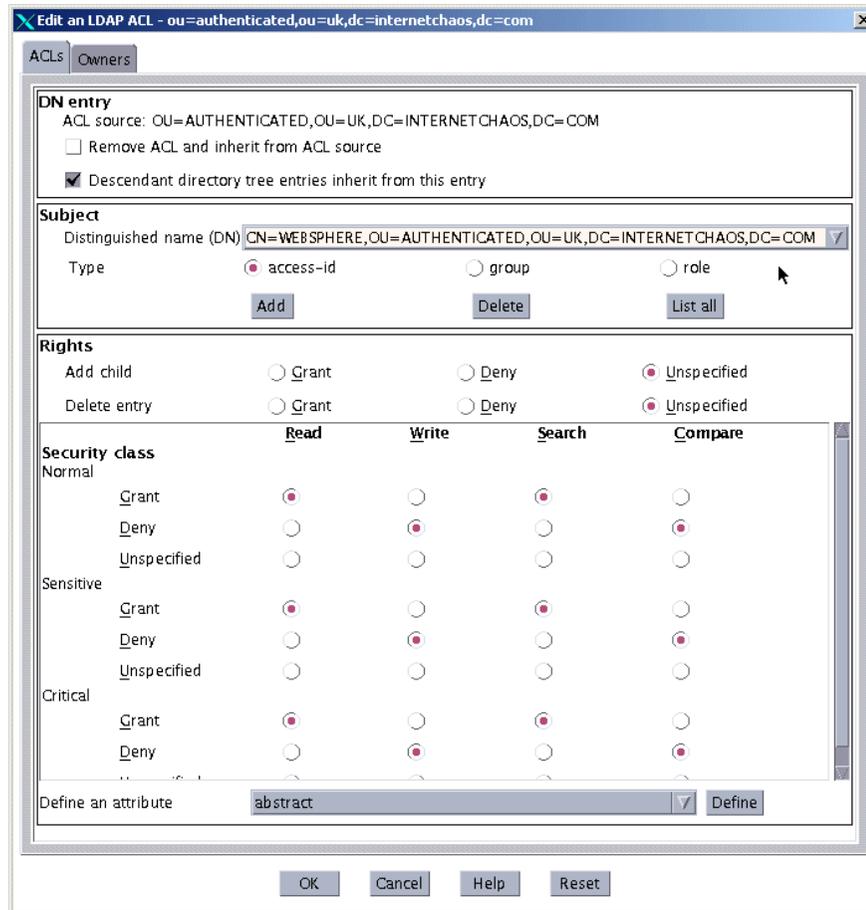


Figure 11-49 LDAP ACL settings

4. The new entry will now appear in the Subject drop-down **Distinguished Name (DN)** menu. Select **CN=ANYBODY** once more and click **Delete**; this will remove the entry that, by default, is currently allowing anonymous connections to view the entries under the *ou=authenticated* organizational unit.
5. To enable the entities *cn=websphere* found under the *ou=authenticated* organizational unit to gain access to the LDAP Directory and their name space, grant the following privileges:

In the **Rights** section, deny the privilege to **Add a child** and **Delete an entry**. For the Security class, only **Grant Read** and **Search** permissions. Deny all others access. Do this for **Normal**, **Sensitive** and **Critical** data types.

6. Click **OK** when you are done.

Restricting access to the admin organizational unit (OU)

We also created a third optional organizational unit, to hold a number of users responsible for maintaining the actual underlying LDAP Directory.

The steps involved in restricting anonymous access to the admin organizational unit are identical to those documented for the authenticated organizational unit, the exception being that the Relative Distinguished Name (RDN) `ou=admin` is alternatively specified in place of the `ou=authenticated` RDN.

If you are going to create such an organizational unit for LDAP administration purposes, you will also have to grant the group creation and modification rights at the respective RDN in the LDAP hierarchy.

11.6.7 SSL and the Netscape iPlanet Alliance Directory Server

In contrast to the IBM SecureWay Directory Server, the iPlanet Directory Server must be configured with an SSL certificate signed by a third-party Certificate Authority (CA). This approach is identical to that used when securing a Web Server. The client (WebSphere), must hold the Certificate Authority (CA) public key as a trusted root. The implementation is shown below in Figure 11-50.

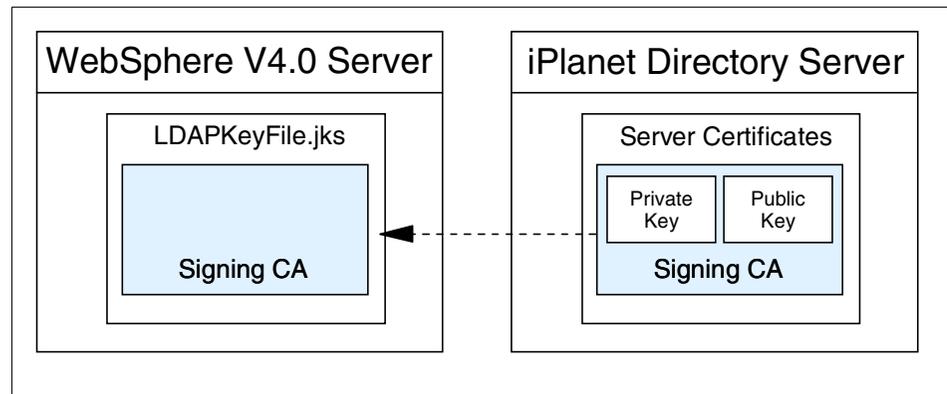


Figure 11-50 Certificates for LDAP secure connection

When WebSphere connects to the LDAP Directory using SSL, the iPlanet public certificate key is sent in response. As this is a certificate chain, signed by the

Certificate Authority (CA) public key resident in WebSphere trusted key database (LDAPKeyFile.jks), SSL handshaking will commence.

The iPlanet Directory instructions provided in this section are only a guide. You should consult the iPlanet official documentation for the definitive configuration procedures endorsed by the vendor.

11.6.8 SSL Certificate creation with iPlanet Directory Server

SSL certificate management is a feature fully integrated into the iPlanet Administrative Console.

1. Start the **iPlanet Console**. The assumption is made that you have previously configured the Directory Server and are using it for WebSphere LTPA authentication, albeit without SSL support.

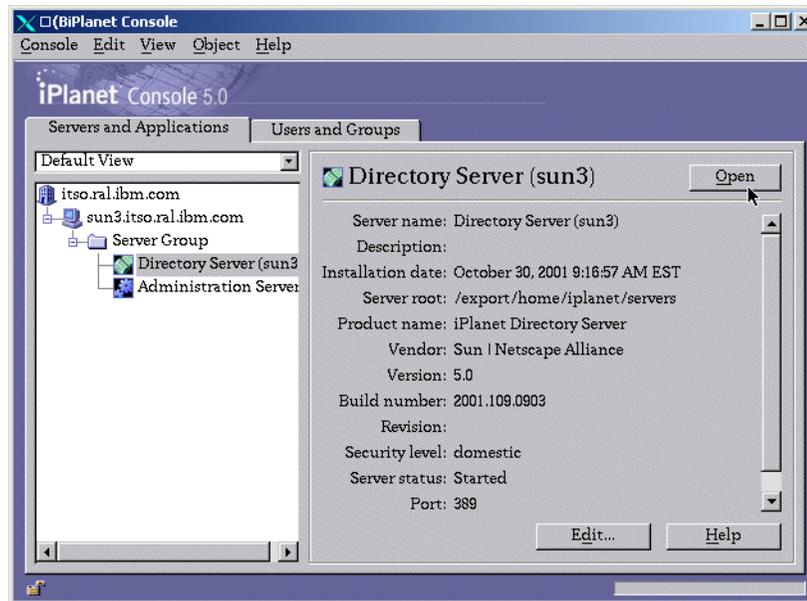


Figure 11-51 iPlanet Console

2. Expand the **Server** and **Applications** topology tree and select your **Directory Server** entity. In the example shown in Figure 11-51, the Directory Server is listed as sun3. Proceed by clicking **Open** in the upper right hand corner of the panel.



Figure 11-52 Directory Server management panel

3. Figure 11-52 shows the next window that you will be presented with. If you are not familiar with the iPlanet Directory Server, you can navigate between the Tasks, Configuration, Directory and Status menus to explore the product.
4. From the **Tasks** menu, select the **Manage Certificates** wizard. This will present you with the window shown in Figure 11-53.



Figure 11-53 Manage Certificates wizard

5. Ensure that you select the **Server Certs** tab and click the **Request** button to start the SSL certificate request process. You will be further prompted to complete the steps detailed by the Certificate Request Wizard. Click **Next** after reading the introduction.
6. Complete the certificate attributes as requested by the Wizard. Care should be taken when specifying the attributes, as the Server name field is often misinterpreted. For example, VeriSign Server IDs are specific to the Server name and must be based on a fully qualified host name. VeriSign also stipulates that the State/Province is completed. You can click the Show DN button, to view the corresponding certificate Distinguished Name.

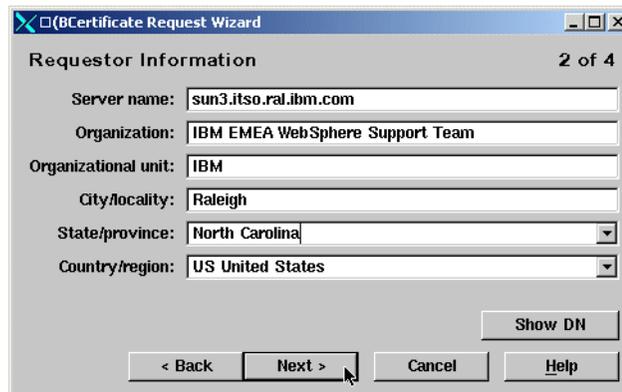


Figure 11-54 Certificate Request Wizard - certificate attributes

7. Click **Next** when you are done.
8. Although you will be creating a Certificate Signing Request (CSR) for submission to your chosen Certificate Authority (CA), you must select a

password to protect the private key of your currently self-signed certificate. Click **Next** when this is completed.

9. Next, save the Certificate Signing Request (CSR) to a file. You can then submit the CSR to the Certificate Authority (CA) of your choice. Typically, this step is completed by cutting and pasting the CSR into the online tool provided by your selected CA. The data will look like this:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB8jCCAVsCAQAwgExCzAJBgNVBAYTA1VTMRAwDgYDVQQREwd0QzI3NzA5MRcw
FQYDVQIQIw50b3J0aCBDYXJvbG1uYTEfMBOGA1UEBxMUMVzZWFyY2ggVHJpYW5n
bGUgUGFyazEKMCIGA1UEChMbRU1FQSBXZWJTcGh1cmUgU3VvcG9ydCBUZWFtMQww
CgYDVQQLLEwNJKk0xIjAgBgNVBAMTGXJzNjE3MDAxLm10c28ucmFsLm1ibS5jb20w
gZ8wDQYJKoZIhvcNAQEBBQADgYOAMIGJAoGBANnpW51HQhbLBb/FV70xYKGTG3wB
dYHrh5HkR8us1+fixRpm3AfwWUF4S4SsCz/fGZ6bT7fuA4k50ymHLz/ZN2Cg/8Z
G9K7qm2yvZIpZYrapLKhNArSVah85bFJKxAUyMW0z01aACaS4Rnkitow1n4NHG7E
Z/0vYUq77kfje159AgMBAAGgADANBgkqhkiG9wOBAQQFAAOBgQAIfcw73CnjDao
3p/AHi2Zxn0VxUznFpGwal1QRRcsP2+B4VZ3mMeCiUZ8APVU5okxYE+C/k3nSY0g
92C+o0YesFGEgXiW4TZ/DJ56/zNWP1S18Wd1VQ9vwt5cKnA3LtNTJFTcDtmh0MNv
aoraXaT0voxAcS3Tgoe1E+5zEmWqHg==
-----END NEW CERTIFICATE REQUEST-----
```

10. Usually, you will receive the response from your chosen Certificate Authority (CA) via e-mail. If the CA has adhered to the RFC 1421 standard, then the response will be returned in Base64-encoded ASCII data format. You can either save the attachment from your e-mail or cut and paste the response to a new file. In our example, we cut and pasted the data to a new file called: iPlanetRES.arm.
11. Before you can install a certificate authenticated by a third-party Certificate Authority (CA), you must ensure that the public certificate of that CA is installed as a trusted authority or root. Such certificates may be one of two types: an Intermediate Certificate Authority or a Trusted Root Certificate Authority. You should ensure that your chosen CA makes their public key available and in a format compatible with the iPlanet LDAP Directory. This intermediate step is shown below in Figure 11-55 and may not be required if your CA certificate is already listed.

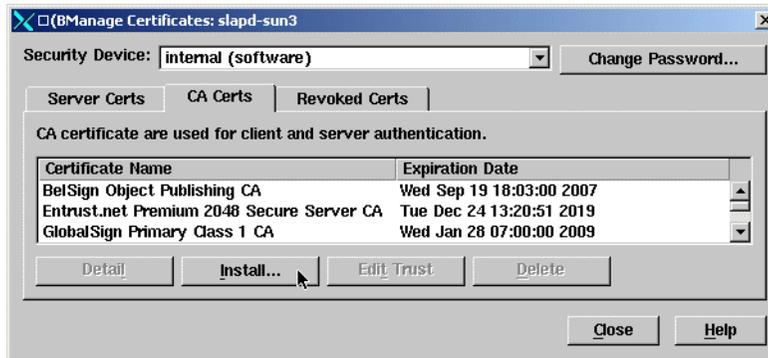


Figure 11-55 Managing certificates

12. Install the Certificate Signing Request (CSR) response from your chosen Certificate Authority (CA) by selecting **Manage Certificate Wizard** from the main iPlanet **Tasks** menu. Ensure that you select the **Server Certs** tab before clicking the **Install** button.
13. In the next window, you need to select the file that contains the certificate CSR response received from the CA. This is depicted in Figure 11-56 below.

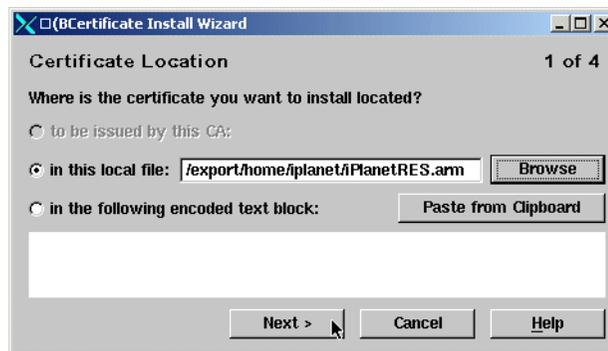


Figure 11-56 Certificate Install Wizard

14. Next, you will be prompted to enter a label name for the certificate; type in server-cert. This is the name that the iPlanet Directory server will later use to reference the certificate when performing SSL handshaking with secure clients.
15. At this point, you will be prompted to enter the Token Password protecting the private key of the certificate. This is the corresponding private key associated with the public key that has now been signed by the third-party Certificate Authority (CA).

16. Click **Done** to complete the certificate installation task.

You can check the status of any certificate stored in the iPlanet Certificate Management center by selecting the concerned certificate and double-clicking it from the main Certificate Management window.

A correctly installed certificate signed by a third-party Certificate Authority (CA) will look similar to that shown in Figure 11-57. In this case, the certificate is issued to *sun3.itso.ral.com* and is signed by the *Thawte Test CA Root*.



Figure 11-57 Certificate details

11.6.9 Modifying iPlanet to support SSL/LDAPS

After successfully installing a certificate signed by a third-party Certificate Authority (CA) into the integrated iPlanet Certificate Management Center, the iPlanet Directory server must be configured to support LDAPS, the SSL secure version of the LDAP protocol.

1. Start the iPlanet Administrative Console and select the **Configuration** menu associated with the LDAP Directory instance for which LDAPS is to be enabled, as illustrated in Figure 11-58.

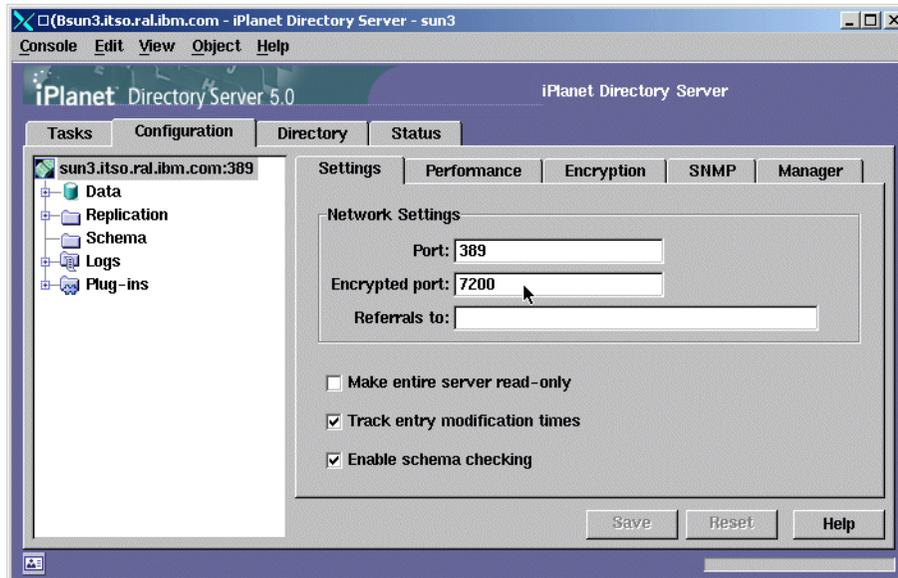


Figure 11-58 Directory server configuration - settings

2. From the **Settings** tab of the Configuration window, define the **Encrypted Port** number that will be used to service secure LDAPS requests. If you are running the LDAP daemon (slapd) as any user other than root on Unix, the port number must be above 1024. In the example shown in Figure 11-58, the Encrypted Port is set to **7200**. Note that although this figure is valid, the default port number for the LDAPS protocol is in fact 636. Adhering to the default port number is perhaps of more concern, if you are using the LDAP Directory server for other functions within your organization.
3. Next, select the **Encryption** tab from the Configuration window.

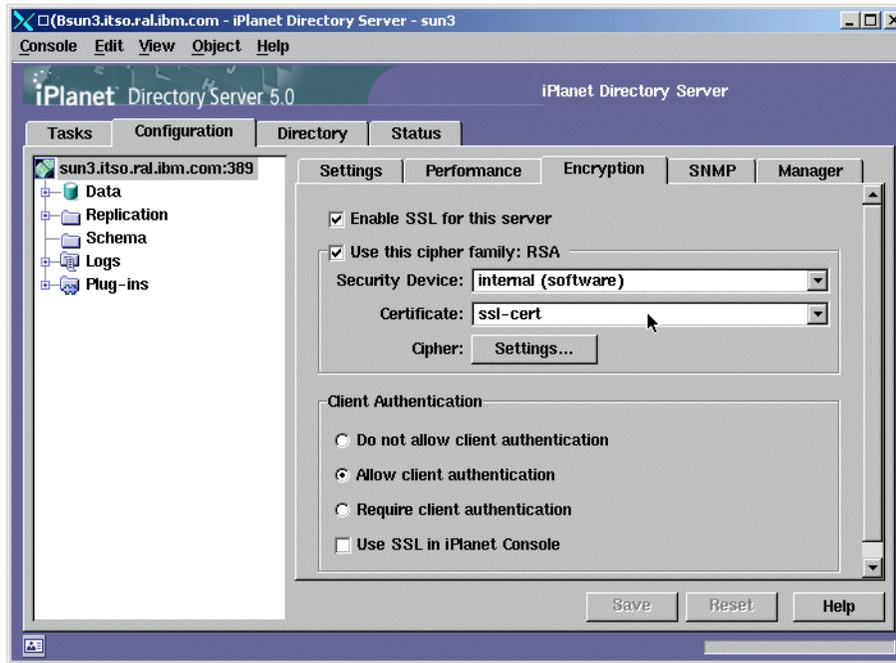


Figure 11-59 Directory Server Configuration - Encryption

4. Set the Certificate field to reference the certificate label name previously created when using the iPlanet Certificate Management Wizard; the choice is only possible from those certificates listed in the drop-down menu. To restrict the SSL encryption strength to a particular algorithm, launch the Cipher **Settings** submenu. Bear in mind that each SSL peer must support the same encryption cipher suite to be able to establish an encrypted session. Do not select the Require client authentication box under the Client Authentication settings, as we have not placed a client certificate into the iPlanet Certificate Management center.
5. Once this is completed, click **Save** and proceed to restart the iPlanet LDAP Directory Server.

11.6.10 The iPlanet CA-signed certificate

The following intermediate step must be completed prior to configuring secure communication between WebSphere and iPlanet.

1. Launch the IBM JSSE ikeyman utility that ships under the WebSphere *bin* directory. On Unix systems this is invoked by running the **ikeyman.sh** script.

7. The act of introducing a new certificate into the key database will prompt you to specify a label name or alias for the entry: type in `thawtetestca`. On completing the prompt, the label name will be listed under the Signer Certificates menu in `ikeyman`.

Note: We suggest that you refrain from using spaces in the label name, even though the default signer certificates of some well known Certificate Authorities include spaces.

8. After checking that the certificate has successfully been installed as a trusted signer, close the key database and quit `ikeyman`.

11.6.11 Modifying WebSphere to support LDAPS with iPlanet

The following steps show how to configure WebSphere to support LDAPS (LDAP over SSL).

1. Start the WebSphere administrative console and launch the Security Center. Make sure WebSphere Global Security is enabled on the General tab, then select the **Authentication** tab to view the Lightweight Third Party Authentication (LTPA) and the LDAP settings, as shown in Figure 11-60.

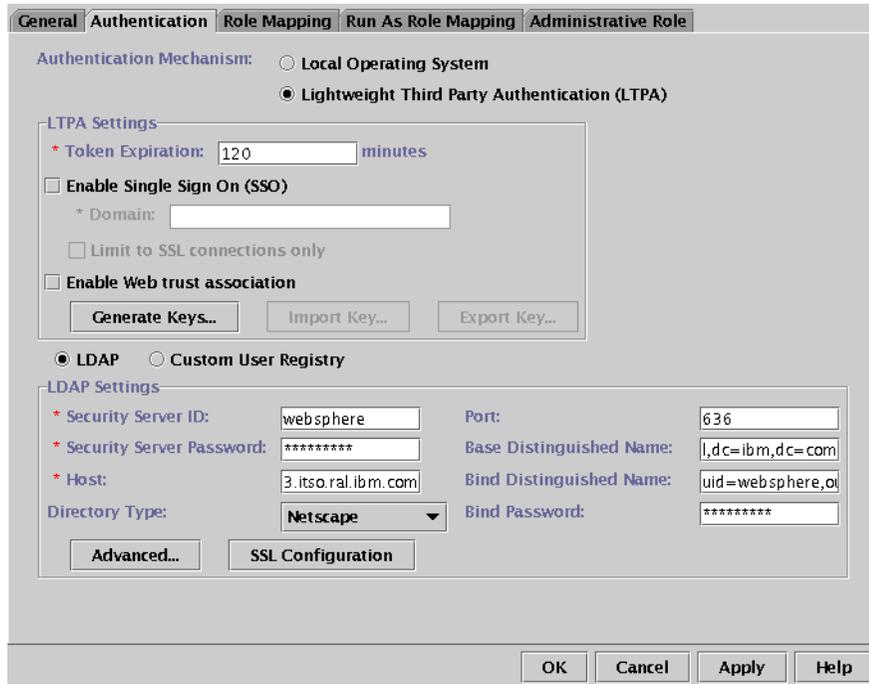


Figure 11-60 WebSphere Global Security - Authentication

2. Apart from the SSL configuration parameters, which we will cover in the next step, and the LDAP port number, all of the settings should remain unmodified as per the non-secure LDAP authentication scheme. The settings and expected values are discussed in detail below:
 - i. **Security Server ID:** specify the user you have created in your LDAP implementation that will represent the WebSphere Server identity, in our example: websphere. Do not use the LDAP cn=root entity under any circumstances.
 - ii. **Security Server Password:** specify the corresponding password for the Security Server user; in this example the password is: websphere.
 - iii. **Host:** this is fully qualified host name of the LDAP Directory server; our host was sun3.itso.ral.ibm.com.
 - iv. **Directory Type:** select **Netscape** on this occasion. Note that if you modify any of the filtering rules, the Directory Type will change to **Custom** even though Netscape remains the LDAP Directory of your choice.
 - v. **Port:** specify **636** which corresponds to TCP/IP port listening for SSL enabled LDAP queries on the remote iPlanet LDAP Directory.

- vi. **Base Distinguished Name:** set this to the entry point into your LDAP Directory Server where WebSphere will search for authenticated users, for example: `dc=itso,dc=ra1,dc=ibm,dc=com`.
- vii. **Bind Distinguished Name:** specify the fully qualified Distinguished Name of the LDAP user that has sufficient privileges to search and authenticate WebSphere users in the LDAP Directory Server. If you have restricted anonymous searches on the WebSphere user name space in your LDAP Directory, this field must be completed regardless of whether or not the user is the same as that specified for the Security Server ID. The Bind Distinguished Name we have used was: `uid=websphere,ou=administrators,dc=itso,dc=ra1,dc=ibm,dc=com`.
- viii. **Bind Distinguished Password:** specify the corresponding password for the Bind Distinguished user; our password was `websphere`.

Note: Care should be taken when specifying the LDAP settings, as the Security Server ID and Base/Bind Distinguished Names are often misinterpreted.

- 3. At this point, *do not* click **Apply** or **OK**.
- 4. Proceed to the LDAP SSL configuration menu by selecting the **SSL Configuration** button. Figure 11-61 shows the resulting window.

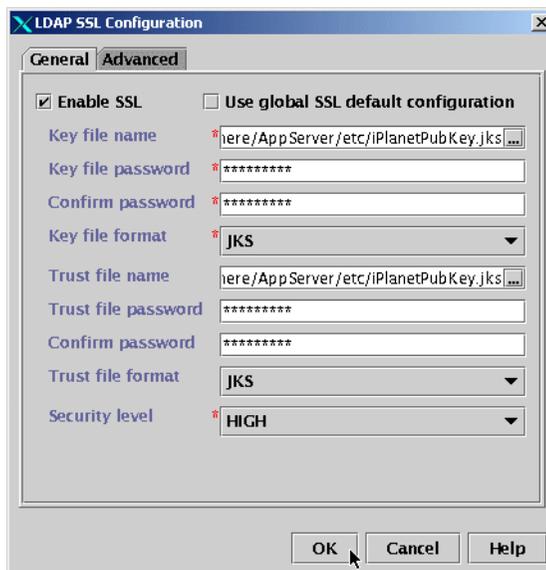


Figure 11-61 LDAP SSL configuration

5. Select the **Enable SSL** box and complete the fields that follow to reference your newly created Java Key Store (JKS) compliant key database.
 - i. **Key file name:** specify the fully qualified file name of the Java Key Store (JKS) key database previously created. In our example, this is set to `/usr/WebSphere/AppServer/etc/iPlanetPubKey.jks`.
 - ii. **Key file password** and **Confirm password:** state the password used to protect the Java Key Store (JKS) certificate key database above.
 - iii. **Key file format:** ensure that **JKS** is selected.
 - iv. **Trust file name:** potentially you can set this to point at a second Java Key Store (JKS) used for holding trusted certificate keys. However, if you choose not to differentiate between personal keys and trusted keys, and opt to use a single key database for both tasks, this field should be set to the same as the Key file name value. In our example this is set to `/usr/WebSphere/AppServer/etc/iPlanetPubKey.jks`.
 - v. **Trust file password** and **Confirm password:** state the password used to protect the Java Key Store (JKS) certificate key database above.
 - vi. **Trust file format:** ensure that **JKS** is selected.
 - vii. **Security level:** setting this to **High** will ensure that the strongest SSL encryption algorithms are used for secure communication. The setting must be compatible with algorithms supported by the SSL peer.
 - viii. Click **OK** when you are done.
6. You will now be returned to the main WebSphere Security Center Authentication window. Double-check the LDAP Settings and click **OK**.
7. If your configuration proves successful, SSL communication will be established between WebSphere and the iPlanet LDAP Directory server at this point. In this case, you can click **Apply** in the WebSphere Security Center window and then proceed to restart WebSphere for the changes to be included in the next runtime.

If, upon clicking **OK**, the configuration fails with an error message and exception, double-check all parameters and certify that the remote iPlanet Directory Server is functioning. Do not click **Apply** or terminate the Admin Client GUI at this point, as you will lose the ability to undo the changes you made.

Future launches of the WebSphere Administration Console will now ask you to log on with the following prompt shown in Figure 11-62.



Figure 11-62 Login to WebSphere Administrator's Console

11.6.12 Disabling iPlanet Anonymous LDAP searches

We first introduced the notion of disabling anonymous LDAP searches with the IBM SecureWay LDAP Directory Server. For the same reason, it is envisaged that you may wish to inhibit such searches if you have installed the iPlanet Directory Server.

1. Launch the iPlanet Administrative Console and connect as an authenticated user with sufficient management privileges.
2. Expand the **Server** and **Applications** topology tree and select your **Directory Server** entity. When you initially created the Directory at install time, iPlanet created an initial Distinguished Name (DN) based on your fully qualified server host name. As such, the example demonstrated in this subsection uses: `dc=itso,dc=ra1,dc=ibm,dc=com`.

Figure 11-63 shows the two additional organization units *wasusers* and *administrators* found below `dc=itso,dc=ra1,dc=ibm,dc=com`. Here, the members of *ou=wasusers* are listed in the right-hand pane.

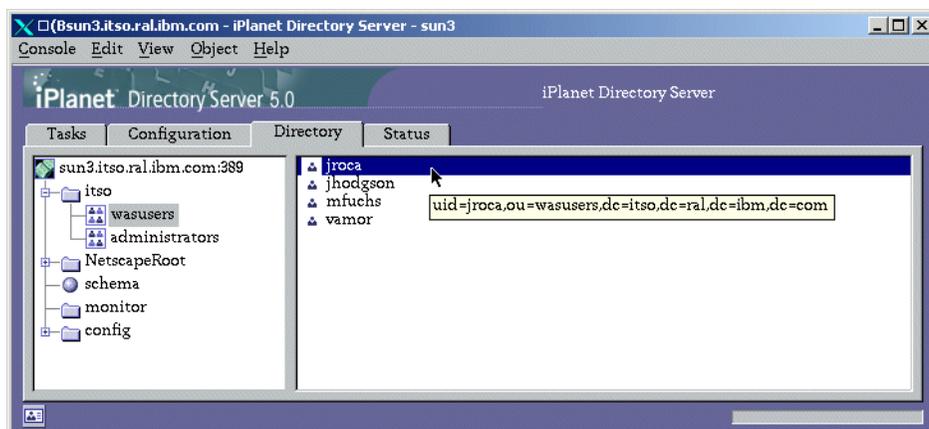


Figure 11-63 Browsing the directory

- Proceed by selecting the **dc=itso** Relative Distinguished Name (RDN), shown in the above figure as a closed folder, and hold down the right mouse button, then select **Set Access Permissions**. Supplement accordingly with your own values.

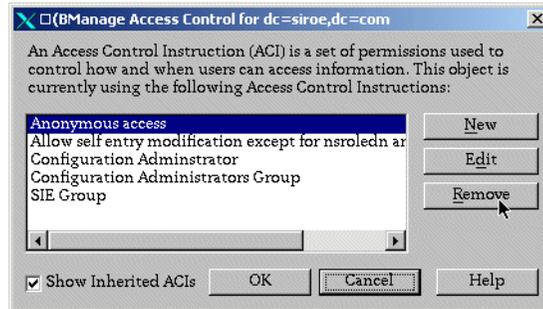


Figure 11-64 Access Control

- By default, if you have not yet modified the Access Control Instructions (ACI) for your selected Distinguished Name (DN), you will find the presence of the Anonymous access ACI. It is this ACI, as shown above in Figure 11-64, that permits anonymous searches of the iPlanet LDAP Directory. Select the **Anonymous access** ACI and click **Remove**.
- Next, you must create a new Access Control Instruction (ACI) to grant the WebSphere administrative user the right to search the LDAP registry. This will allow WebSphere to perform user authentication. Click **New** and enter the ACI name WAS-Access (or a name of your choice).

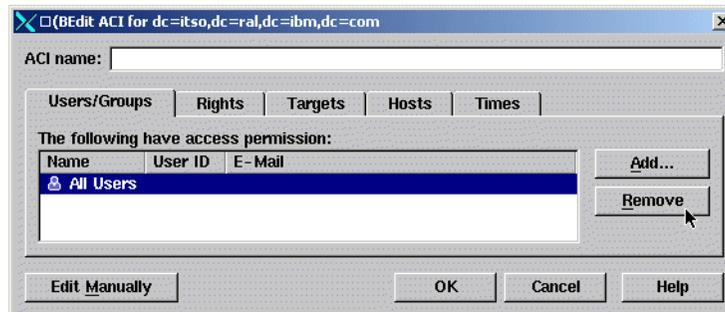


Figure 11-65 Editing the Access Control Instruction

- In the subsequent window, as shown above in Figure 11-65, delete the **All Users** entry from the Users/Groups section. This is necessary, as we want to restrict LDAP access to the WebSphere administrative user only.
- Click **Add** to specify a new self-defined user.

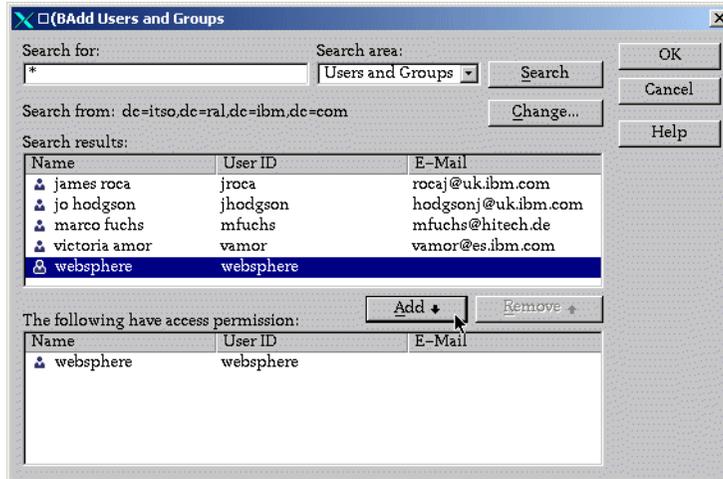


Figure 11-66 Adding a User/Group

8. In the resulting window, enter a wildcard (*) in the Search field before clicking the **Search** button. This action will then bring in all of the entries previously populated under in the Directory. Select the **websphere** user and add it to the list of users to have access permission. Click **OK** when you are done.
9. The websphere user will now appear as the user listed as having access permissions for the new Access Control Instruction (ACI). Select the **Rights** tab and ensure that **Read** and **Search** are selected.

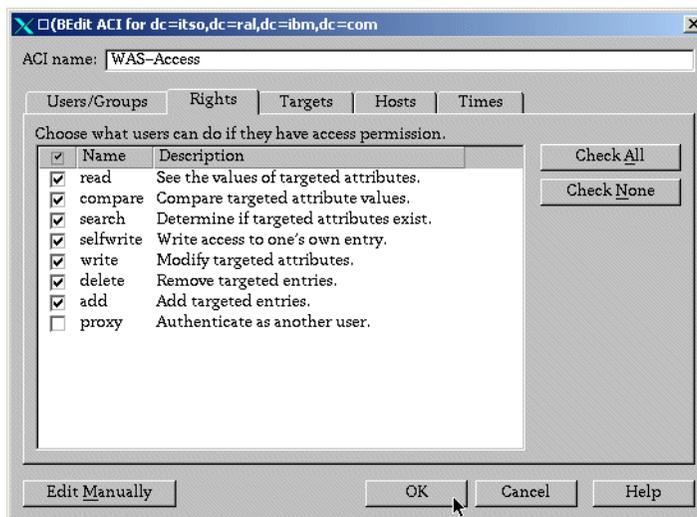


Figure 11-67 Access rights for the ACI

10. Click **OK** when you are done to return to the main Access Control Instruction (ACI) window. The newly created WAS-Access ACI will now be listed. Click **OK**.
11. As the Access Control Instructions (ACI) are inherited, if you select a user from under the wasusers organization unit (ou=wasusers), you will discover that the ACI previously set has been cascaded to that child. It might be that no specific ACIs are set on an individual basis, but if you select the **Show Inherited ACIs** box, the inherited ACIs can be viewed.

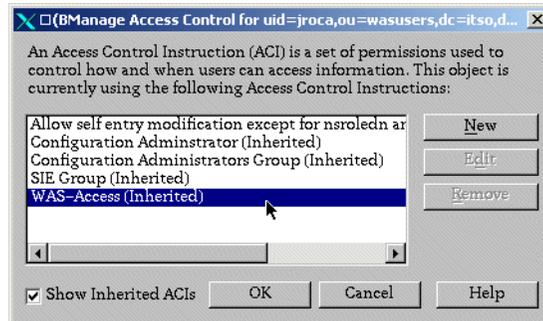


Figure 11-68 Selecting the ACI

You can further override the ACIs of a parent by specifying a particular ACI on a child entity, assuming you hold adequate privileges to do so. For example, you could add an entry for an anonymous search back to a specific user or group of users.

11.6.13 SSL and Lotus Domino LDAP

This subsection documents the steps that must be taken to achieve secure communication between WebSphere V4.0 and Lotus Domino, when Lotus Domino is used as the selected LDAP Directory Server.

We also assume you have previously configured Lotus Domino as a fully functioning LDAP Directory Server with WebSphere, albeit without SSL support, before commencing this section.

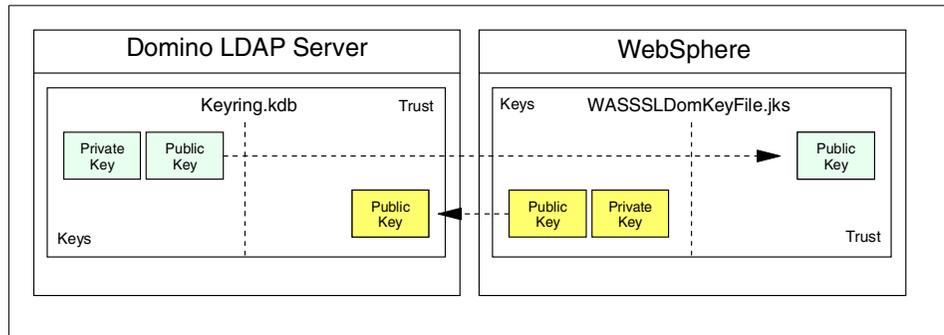


Figure 11-69 Certificates for secure LDAP connection

Figure 11-69 shows the requirements for the SSL certificates involved in establishing secure communication between WebSphere and Lotus Domino. Note that the public certificate keys are exchanged between adjacent peers.

Creating a self-signed certificate for Lotus Domino

The following steps will show you how to create a self-signed certificate for Lotus Domino.

There are a number of ways to create a valid SSL certificate for establishing secure communication between WebSphere and the Lotus Domino LDAP Server. In this section, we will use the integrated Domino certificate management utility.

1. Launch your Notes client, making sure you have adequate administration privileges. Select your chosen server and open the certserv.nsf database. This will allow you to create a new public/private self-signed certificate key pair.

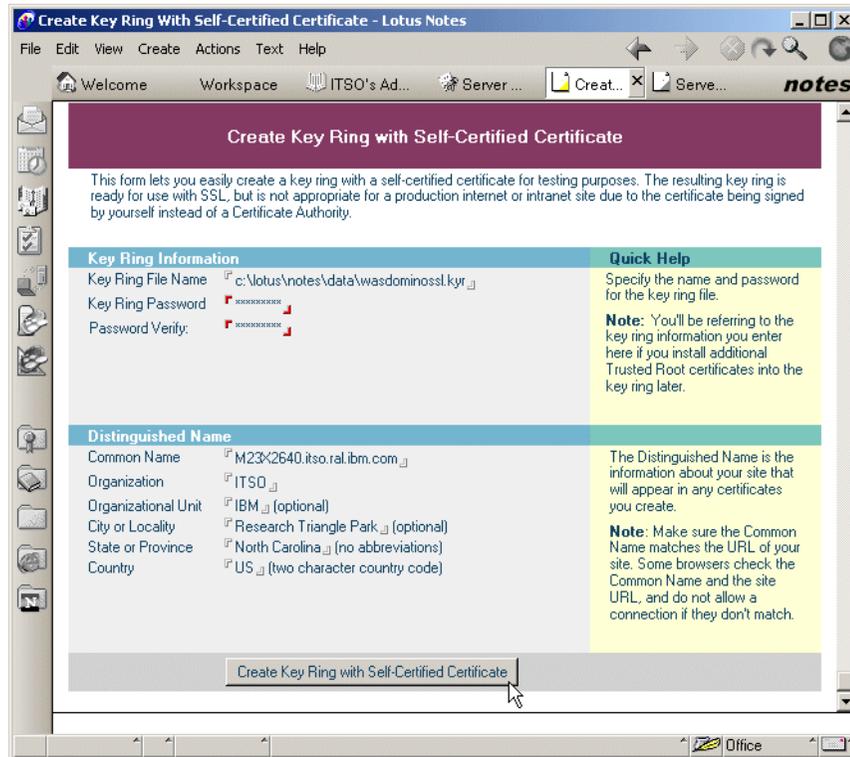


Figure 11-70 Creating a self-signed certificate in Domino

2. Complete the following fields once at the Create Key Ring with Self Certified Certificate template, as shown in Figure 11-70:

- i. **Key Ring File Name:** specify the fully qualified file name of the certificate key database file, in our example:
c:\lotus\notes\data\wasdominoss1.kyr.
- ii. **Key Ring Password** and **Password Verify:** state the password used to protect the certificate key database above.
- iii. The following options then need to be specified. You may choose to complete all of the optional fields for completeness. Supplement your own values accordingly.

Common name: M23X2640.itso.ral.ibm.com

Organization: ITSO

Organization Unit: IBM

City or Locality: Research Triangle Park

State or Province: North Carolina

Country: US

3. Clicking the **Create Key Ring with self-signed Certificate** button will present you with the certificate summary information as shown in Figure 11-71.

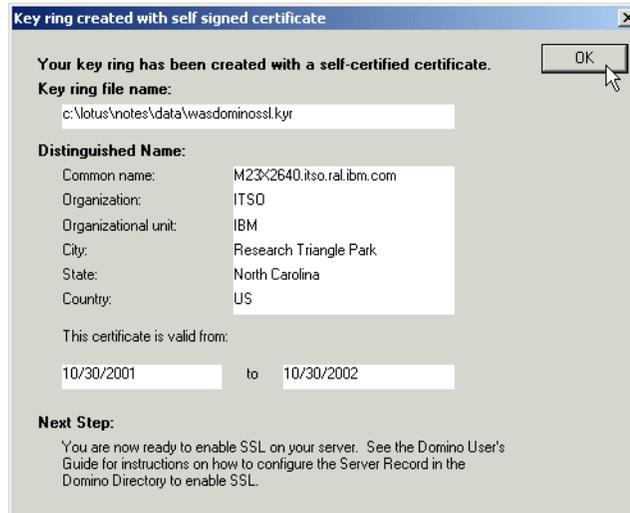


Figure 11-71 Certificate details

4. If the information returned is correct, click the **OK** button in the top right-hand corner of the window. Domino will then create two files: wasdominoss1.kyr and wasdominoss1.sth under C:\Lotus\Notes\Data on your local machine.
5. To enable WebSphere to encrypt secure requests to the Domino Server, we must now extract the associated public certificate key. A good analogy of the public key is an open safe. When the safe is closed, only the private key resident at the LDAP server can unlock the safe and retrieve the data.

The certificate database/keyfile created with Lotus Domino is compatible with the ikeyman tool that ships as part of the IBM GSKIT. In this case, proceed to open the wasdominoss1.kyr keyfile with the ikeyman tool. Note that this is not the JSSE ikeyman tool that ships in the WebSphere *bin* directory.
6. You will find the self-signed certificate key pair, as created in Domino, named KeyPair, under the ikeyman Personal Certificates menu section. Select the certificate and click the **Extract Certificate** button.

7. You will then be prompted to complete the fields, then click **OK** when you are done.

Data type: Base64-encoded ASCII data

Certificate file name: DominoSSLPubCert.arm

Location: /usr/WebSphere/AppServer/etc/domino

8. One step that is optional at this point is to delete all of the trusted root certificates, as listed under the ikeyman Signer Certificates menu. This is possible, as we are concerned only with using self-signed certificates and will not be authenticating against any third-party Certificate Authority (CA).
9. Close the key database and quit ikeyman when you are finished.

Creating a self-signed certificate for WebSphere

It is also acceptable to create a private/public self-signed certificate key pair for the WebSphere peer. In this case, follow these steps:

1. Launch the IBM JSSE ikeyman utility that ships under the WebSphere V4.0 *bin* directory. On Unix systems, this is invoked by running the **ikeyman.sh** script.
2. Create a new Java Key Store (JKS) database to store a self-signed certificate for the WebSphere Application Server peer and the public certificate key from the remote Domino LDAP Server.

From the ikeyman menu bar, select **Key Database File -> New**. In the window that follows, ensure that the key database file is set to support the Java Key Store (JKS) format. Complete the file name and location, then click **OK** when you are done.

Key database type: JKS

File Name: WASSSLDomKeyFile.jks

Location: /usr/WebSphere/AppServer/etc/domino

3. At the Password Prompt window, enter the password of your choice. The more random the password, the higher the password strength. There is no requirement to stash the password to a file with the JSSE ikeyman utility.
4. Optionally, as no requirement exists to authenticate against any of the default trusted root Certificate Authority (CA) certificates installed under the ikeyman Signer Certificates menu, you can choose to delete all of the CA trusted root certificates.
5. From the ikeyman menu bar, select **Create -> New Self-Signed Certificate** to create a new private/public self-signed certificate key pair. The following options, as shown in Figure 11-72 then need to be specified. Supplement your own values accordingly.

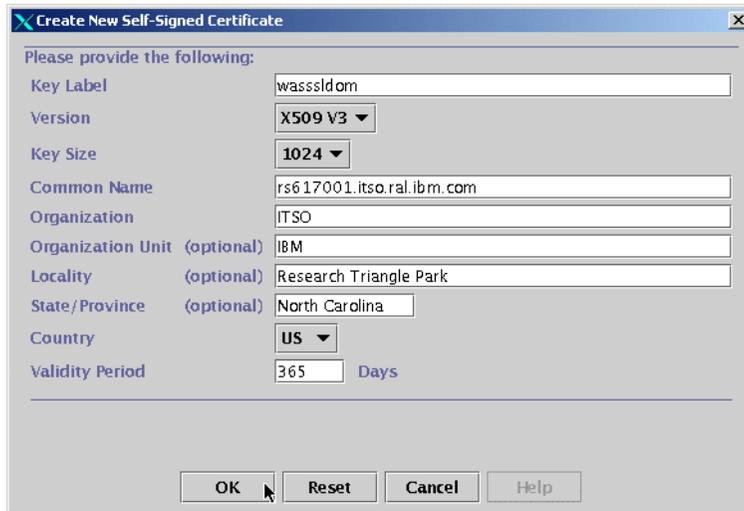


Figure 11-72 Certificate details

6. Click **OK** when you are done. You will then find that the certificate is listed under the ikeyman Personal Certificates menu, by the Key Label name.
7. Extract (not export) the public key of this certificate, as you will need to install this into the certificate database/keyfile used by the Lotus Domino LDAP Directory peer. From the Personal Certificates window, ensure that the certificate just created is selected and click the **Extract Certificate** button. Complete the fields, then click **OK** when you are done.

Data type: Base64-encoded ASCII data

Certificate file name: WASSSLDomPubCert.arm

Location: /usr/WebSphere/AppServer/etc/domino

8. Close the key database and quit ikeyman when you are finished.

Exchanging the public certificate keys

At this point, you will have two certificate key databases/keyfiles and two .arm files containing the associated extracted public certificate keys, one from the Lotus Domino peer and one from the WebSphere Application Server peer.

You must now exchange the public keys between the adjacent peers for secure communication to be possible.

Dealing first with the Domino peer, open the key database originally created with the integrated Domino certificate management utility using the CMS compatible ikeyman tool, as supplied with the IBM GSKIT. The GSKIT ships with the IBM HTTP Server (IHS), the IBM SecureWay Directory Server and is thus supplied on the WebSphere V4.0 install media.

1. Launch the IBM GSKIT ikeyman tool and proceed to open the wasdominossl.kyr certificate database/keyfile (the Domino peer certificate database/keyfile).
2. When prompted, enter the password.
3. Select the ikeyman **Signer Certificates** menu. You may see five trusted root Certificate Authority (CA) signer certificates, if you refrained from deleting them earlier when extracting the Domino public certificate key.
4. Remaining in the Signer Certificates menu, click the **Add** button to import the public certificate key from the WebSphere peer. In this case, you will be prompted with the window. Specify the values as stated below, before clicking **OK**.

Data type: Base64-encoded ASCII data

Certificate file name: WASSSLDomPubCert.arm

Location: /usr/WebSphere/AppServer/etc/domino

5. The action of introducing a new certificate into the certificate database/keyfile will prompt you to specify a label name or alias for the entry. Complete the prompt, type in wasssl. The label name will be listed under the ikeyman Signer Certificate menu.

Note: We suggest that you refrain from using spaces in the label name, even though the default signer certificates of some well known Certificate Authorities include spaces.

6. This concludes the tasks necessary to add the WebSphere public certificate key into the Domino certificate database/keyfile. You can now close the certificate database and quit the ikeyman tool.

The next step is to undertake the reciprocal arrangement for the public certificate key associated with the Domino server, adding it to the WebSphere certificate database/keyfile as a trusted signer.

1. Once more, launch the IBM JSSE ikeyman utility that ships under the WebSphere V4.0 *bin* directory. On Unix systems this is invoked by running the **ikeyman.sh** script.
2. Proceed to open the Java Key Store (JKS) WASSSLDomKeyFile.jks certificate database/keyfile (the WebSphere peer certificate database/keyfile).
3. When prompted, complete the password.
4. Select the ikeyman Signer Certificates menu. You may see the trusted root Certificate Authority (CA) signer certificates, if you refrained from deleting them earlier when extracting the WebSphere public certificate key.
5. Remaining in the Signer Certificates menu, click the **Add** button to import the public certificate key from the Domino peer. In this case, you will be prompted with the window. Specify the values as stated below, before clicking **OK**.

Data type: Base64-encoded ASCII data

Certificate file name: DominoSSLPubCert.arm

Location: /usr/WebSphere/AppServer/etc/domino

6. The action of introducing a new certificate into the certificate database/keyfile will prompt you to specify a label name or alias for the entry. Complete the prompt, type in `dominoss1`. The label name will be listed under the ikeyman Signer Certificate menu.

Note: We suggest that you refrain from using spaces in the label name, even though the default signer certificates of some well known Certificate Authorities include spaces.

This concludes the tasks necessary to add the Domino public certificate key to the WebSphere certificate database/keyfile. You can now close the certificate database and quit the ikeyman tool.

Modifying the Domino LDAP Directory Server to support SSL

After successfully generating and exchanging the public certificate keys, both Domino and WebSphere must be configured to support SSL. In this section, we specifically look at the tasks required to enable SSL on the Domino server.

1. Ensure that you have placed both the certificate database/keyfile and the associated stash file into the Domino data directory, on your Lotus Domino Server. In our case, this was `c:\lotus\domino\data`.
2. Start the Domino Administration client, ensuring that you are logged in as a user with adequate administrative privileges. Then open the corresponding Domino Server that will be SSL-enabled, by selecting the server from the left bookmark pane.

3. Click the **Configuration** tab and select **Server -> Current Server Document**.
4. Click **Ports -> Internet Ports** to view and modify the LDAP specific port settings. The respective fields are illustrated below in Figure 11-73.

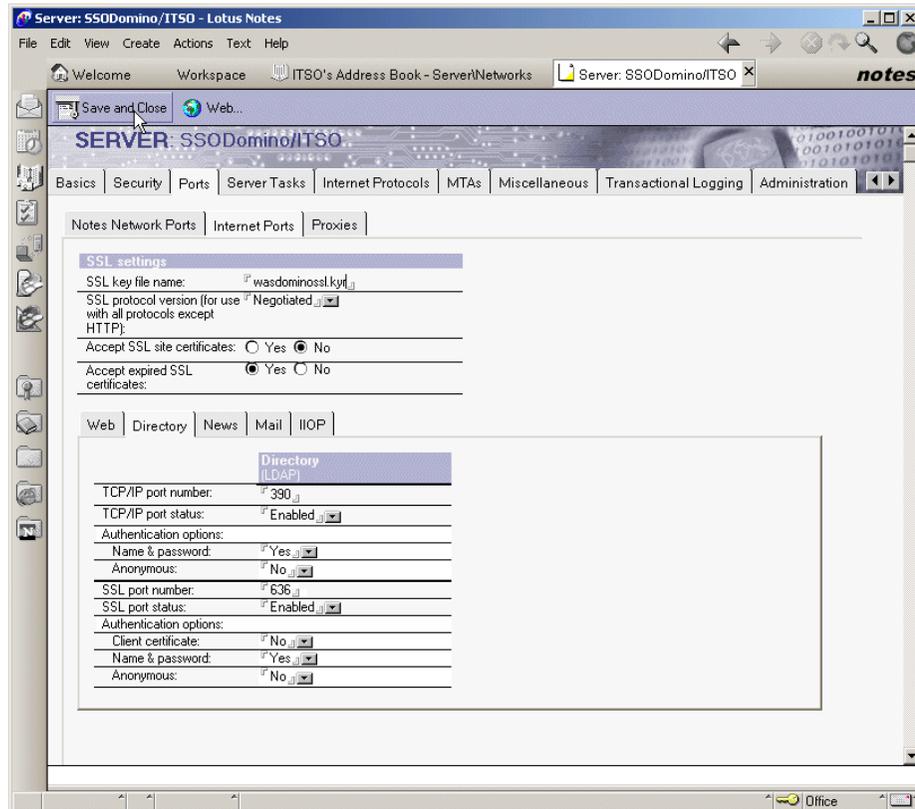


Figure 11-73 Configuring Lotus Domino for LDAPS

5. Modify the parameters to enable SSL support. The settings and expected values are discussed in detail below:
 - i. **SSL key file name:** specify the file name of the certificate database previously created containing the private/public certificate key pair for the Domino server and the public certificate key from WebSphere, for example: wasdominoss1.kyr.
 - ii. **SSL protocol version:** select **Negotiated** to enable the Domino LDAP Server to negotiate the encryption cipher algorithm with the remote WebSphere Application Server peer.

- iii. **Accept SSL site certificates:** there is no need to enable this function when authenticating WebSphere LDAP requests.
 - iv. **Accept expired SSL certificates:** you may choose to select this option if either of your certificates has expired. However, typically you will want to refrain from honoring expired SSL certificates. In our case, this option was set to **No**.
 - v. **SSL port number:** select **636** which is the Internet standard for secure LDAP communication. You may choose any port that is not in use, but you need to ensure that you set the same value when configuring the WebSphere peer. On Unix, only root has access to the ports below 1024 by default.
 - vi. **SSL port status:** select **Enabled** to turn SSL support on at the next restart of the Domino Server.
 - vii. **Client certificate:** you may choose to allow Client certificates. However, because we have exchanged the Domino public certificate key with the WebSphere peer, this is not an absolute necessity; select **No**.
 - viii. **Name and password:** we will be specifying an LDAP Bind Name and password when connecting to the Domino Directory, so specify **Yes**.
 - ix. **Anonymous:** to stop anonymous users from searching the LDAP database, set this option to **No** for both the standard non-secured port and secured port.
6. Once you have completed the fields, click **Save** and **Close** for the modifications to be saved.
 7. You must now restart the Domino Server for the modifications to be included in the runtime. In this case, launch the Domino Administrator client, or launch the Domino Server console from the command line.
 8. Click the Servers icon, then **Expand All Servers -> Select SSODomino/ITSO** (or your Domino server instance). You can then select the Server in the menu bar, **LDAP Server**, as shown in Figure 11-74.

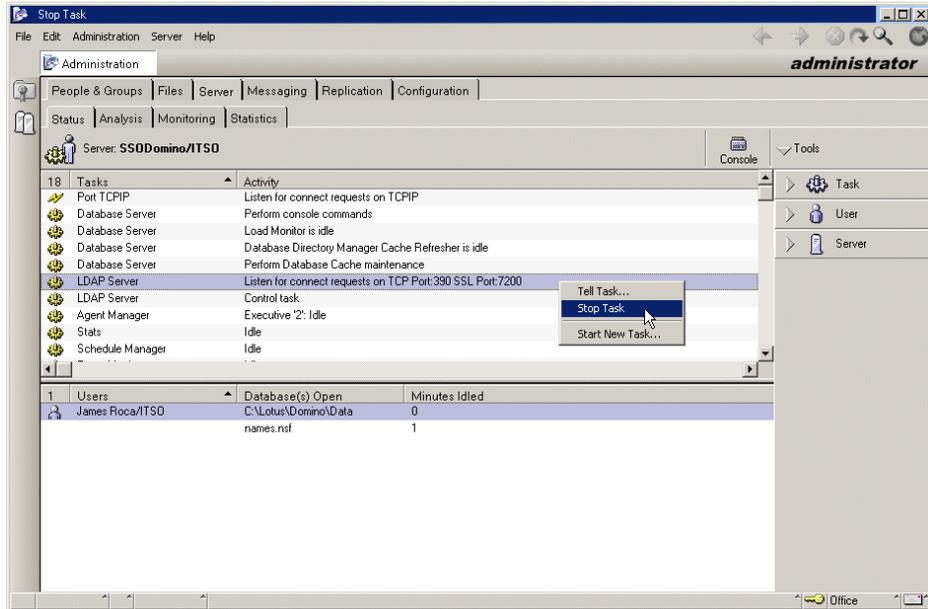


Figure 11-74 Stopping the Domino LDAP server

- From the Task navigation bar on the right-hand side, select **Start**; this will launch the menu shown in Figure 11-75, where you should select and start the LDAP Server.

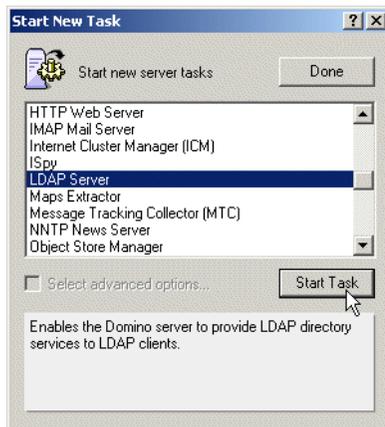


Figure 11-75 Starting a new task in Domino

Modifying WebSphere to support SSL

The final step in implementing LDAPS support between WebSphere and Lotus Domino is to actually configure the SSL setting within WebSphere. The steps have already been introduced with the IBM SecureWay LDAP Directory Server and the iPlanet LDAP Directory Server, but are nevertheless detailed here for completeness.

1. Start the WebSphere Administrative Console and launch the **Security Center**. Select the **Authentication** tab to view the Lightweight Third Party Authentication (LTPA) and the LDAP settings, as shown below in Figure 11-76.

The screenshot shows the 'Authentication' tab of the Security Center configuration. It is divided into two main sections: LTPA Settings and LDAP Settings.

Authentication Mechanism: Local Operating System, Lightweight Third Party Authentication (LTPA)

LTPA Settings:

- * Token Expiration: 120 minutes
- Enable Single Sign On (SSO)
 - * Domain: itso.ral.ibm.com
 - Limit to SSL connections only
- Enable Web trust association
- Buttons: Generate Keys..., Import Key..., Export Key...

LDAP Settings: LDAP, Custom User Registry

LDAP Settings:

- * Security Server ID: wasadmin
- * Security Server Password: *****
- * Host: 0.itso.ral.ibm.com
- Directory Type: Domino 5.0
- Port: 636
- Base Distinguished Name: [empty]
- Bind Distinguished Name: wasadmin
- Bind Password: *****
- Buttons: Advanced..., SSL Configuration

At the bottom of the dialog are buttons for OK, Cancel, Apply, and Help.

Figure 11-76 Security Center - Authentication

2. Apart from the SSL Configuration parameters, which we will cover in the next step, and the LDAP port number, all of the settings should remain unmodified as per the non-secure LDAP authentication scheme. The settings and expected values are discussed in detail below:
 - i. **Security Server ID:** specify the user you have created in your LDAP implementation that will represent the WebSphere Server identity, in our example: wasadmin.

- ii. **Security Server Password:** specify the corresponding password for the Security Server user; we have used **wasadmin** for a password.
- iii. **Host:** this is the fully qualified host name of the LDAP Directory server: M26X2640.itso.ral.ibm.com.
- iv. **Directory Type:** select **Domino** on this occasion. Note that if you modify any of the filtering rules, the Directory Type will change to **Custom** even though Domino remains the LDAP Directory of your choice.
- v. **Port:** specify **636** which corresponds to TCP/IP port listening for SSL enabled LDAP queries on the remote SecureWay LDAP Directory.
- vi. **Base Distinguished Name:** this is usually set to the entry point into your LDAP Directory Server where WebSphere will search for authenticated users. However, when Domino is the selected Directory Server, this field can be left blank.
- vii. **Bind Distinguished Name:** specify the name of the remote Domino LDAP user that has sufficient privileges to search and authenticate WebSphere users. If you have restricted anonymous searches on the WebSphere user name space in your LDAP Directory, this field must be completed regardless of whether or not the user is the same as that specified for the Security Server ID. The Bind Distinguished Name we have used was: `wasadmin`.
- viii. **Bind Distinguished Password:** specify the corresponding password for the Bind Distinguished user; our password was `wasadmin`.

Note: As we disallowed anonymous access, the Bind Distinguished Name and Bind Password fields must be completed.

- 3. At this point do *not* click **Apply** or **OK**.
- 4. Proceed to the LDAP SSL Configuration menu by selecting the **SSL Configuration** button.
- 5. Check the **Enable SSL** box and complete the fields that follow to reference your newly created Java Key Store (JKS) compliant certificate key database.
 - i. **Key file name:** specify the fully qualified file name of the Java Key Store (JKS) key database previously created. In our example this is set to `/usr/WebSphere/AppServer/etc/WASSSLDomKeyFile.jks`.
 - ii. **Key file password** and **Confirm password:** state the password used to protect the Java Key Store (JKS) certificate key database above.
 - iii. **Key file format:** ensure that **JKS** is selected.

- iv. **Trust file name:** potentially, you can set this to point at a second Java Key Store (JKS) used for holding trusted certificate keys. However, if you choose not to differentiate between personal keys and trusted keys, and opt to use a single key database for both tasks; this field should be set to the same value as the Key file name. In our example, this is set to `/usr/WebSphere/AppServer/etc/WASSSLDomKeyFile.jks`.
 - v. **Trust file password** and **Confirm password:** state the password used to protect the Java Key Store (JKS) certificate key database above.
 - vi. **Trust file format:** ensure that **JKS** is selected.
 - vii. **Security level:** setting this to **High** will ensure that the strongest SSL encryption algorithms are used for secure communication. The setting must be compatible with algorithms supported by the SSL peer.
 - viii. Click **OK** when you are done.
6. You will now be returned to the main WebSphere Security Center Authentication window. Double-check the LDAP settings and click **OK**.

If your configuration proves successful, SSL communication will be established between WebSphere and the SecureWay LDAP Directory server at this point. In this case, you can click **Apply** in the WebSphere Security Center window and then proceed to restart WebSphere for the changes to be included in the next runtime.

If, upon clicking **OK**, the configuration fails with an error message and exception, double-check all parameters and certify that the remote LDAP Server is indeed running. Do not click **Apply** or terminate the WebSphere Admin Client console at this point if you experience an error, as you will lose the ability to undo the changes that you made.



Part 3

End-to-end security solutions



Security in Patterns for e-business

This chapter discusses the security considerations within end-to-end solutions in the context of the Patterns for e-business. First, a short overview of the Patterns of e-business will introduce the basic elements of this strategy, focusing especially on security. The last sections will provide design ideas for overall security in an end-to-end solution.

12.1 Patterns for e-business

The primary goal of the Patterns for e-business is to show the strategy of reuse for e-business application design.

Figure 12-1 describes how the different patterns are organized and built on top of each other. These elements also represent logical steps in the solution design process.

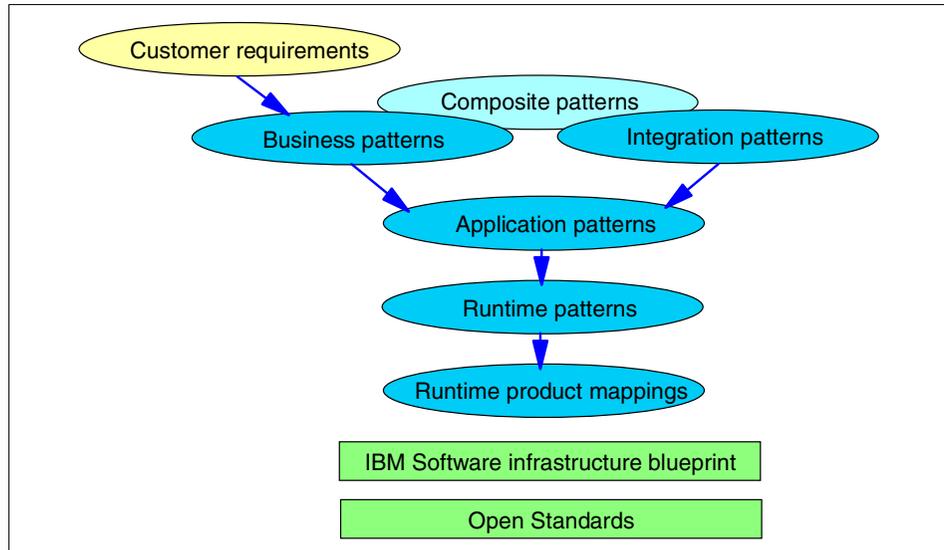


Figure 12-1 Patterns for e-business

The Patterns for e-business define the Integration patterns beside the Business patterns and Composite patterns. Integration patterns integrate multiple applications, multiple modes of access, and multiple sources of information to build one seamless application. Integration patterns are differentiated from Business patterns in that they do not themselves automate specific business problems. These Integration patterns are used within Business patterns to support more advanced functions, or to make Composite patterns possible by allowing the integration of two or more Business patterns.

The Access Integration and Application Integration patterns represent the two types of Integration patterns.

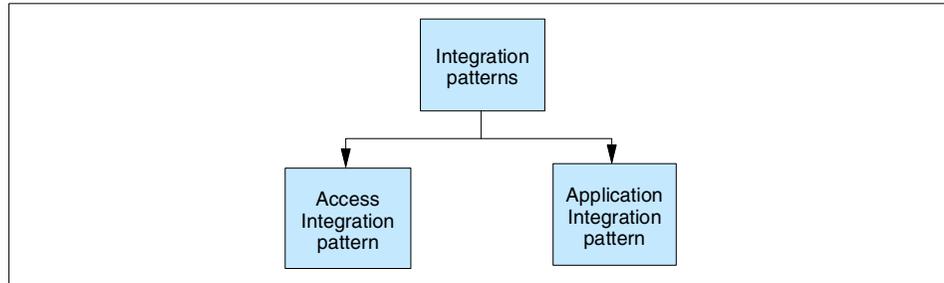


Figure 12-2 Integration patterns

The Access Integration pattern can be thought of as a front-end Integration pattern. The Access Integration pattern is the main focus of this chapter.

12.2 Access Integration pattern

The Access Integration pattern describes the services and components commonly required to provide users with consistent, seamless, secure, device-independent access to relevant applications and information. Access Integration patterns are useful when:

- ▶ Users need access to multiple applications and information sources through a Single Sign-On and application-independent security context.
- ▶ Applications need to be accessible using multiple device types, including fat clients, browsers, voice response units, mobile devices and PDAs.
- ▶ There is a requirement to provide a common look and feel to a collection of applications or to aggregate result sets from discrete applications in a business process.
- ▶ The user wishes to customize the choice of applications and how they are presented.
- ▶ The business wishes to target information and applications to a specific user or group.

Access Integration patterns observed in practice are composed of the following services:

- ▶ Presentation
- ▶ Personalization
- ▶ Security and Administration
- ▶ Pervasive Device Support

The benefits of Access Integration are often best realized when these services are combined. However, in this chapter we will focus on the Security and Administration service of the Access Integration pattern. So, before going any further, let us introduce the Security and Administration service.

The Security and Administration service enables users to access multiple applications and information sources using an integrated security model and a single set of security credentials.

We are all familiar with having to remember multiple user IDs and passwords to access different applications. Applications frequently have different user ID and password format requirements and expiration rules. These differences in authentication requirements inconvenience users by requiring them to remember multiple user IDs and password combinations and by forcing them to sign on multiple times to access different systems. In addition, some systems may use authentication techniques such as certificates or biometrics that are not based on a user ID and password. The primary business driver for this service is to eliminate these user inconveniences while continuing to protect the security of enterprise data and applications.

Under these conditions, a key requirement is a Single Sign-On, so that a user logs on once to gain access to all the appropriate applications and data sources. Another key requirement is access management, which limits access based on a user profile and the content access policies of the enterprise.

Now that we have discussed the Integration patterns found in the Patterns for e-business, specifically the Access Integration pattern, we move to the next logical step in the solution design process. The Application patterns are the next logical step to discuss within Patterns for e-business.

12.2.1 Application patterns

An Application pattern shows the principal layout of the application, focusing on the shape of the application, the application logic, and the associated data.

Application patterns for Access Integration are composed of the services discussed above. Based on the specific application needs, you mix and match these services to facilitate consistent and seamless access to multiple applications.

Four commonly observed Application patterns for Access Integration are shown in Figure 12-3 on page 337.

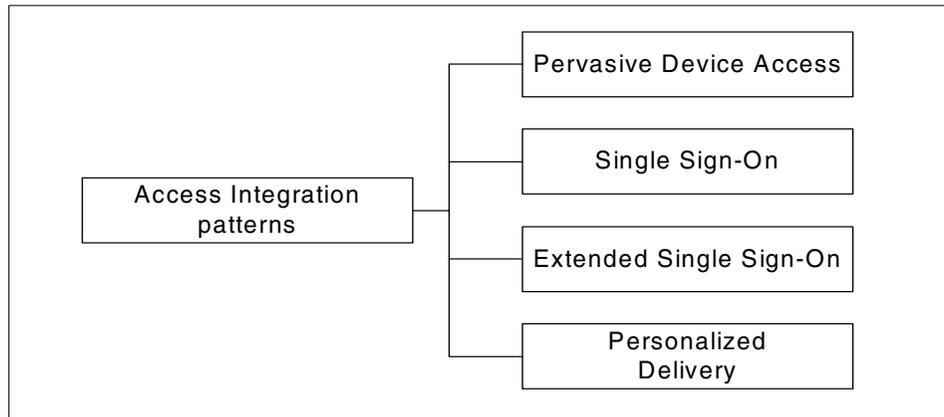


Figure 12-3 Application patterns for Access Integration

These patterns are not mutually exclusive. It is very likely that more than one of these can be found in a single solution. However, for our discussion, we will only focus on the Single Sign-On and the Extended Single Sign-On Application patterns.

For more details on all four of these Application patterns, refer to the redbook *Access Integration Pattern Using WebSphere Portal Server*, SG24-6267 and *Mobile Applications with WebSphere Everyplace Access Design and Development*, SG24-6259.

Single Sign-On Application pattern

This is a basic pattern where the single sign-on functions are performed in the Web tier. This Application pattern includes a Single Sign-On tier, which ensures that the user does not have to log in to the application more than once during a session. Single Sign-On takes over user authentication with other applications once the user has logged in.

The Security and Administration service is used here.

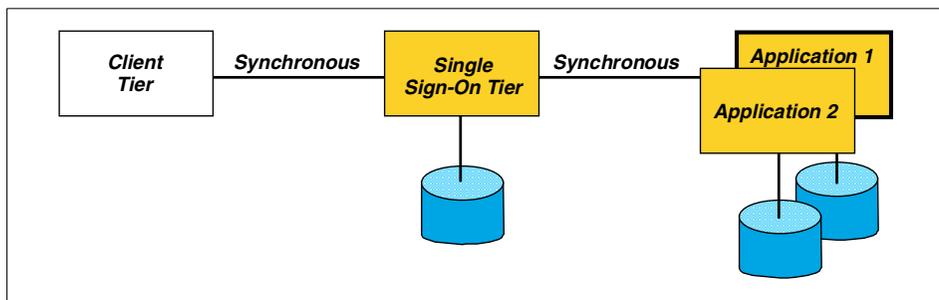


Figure 12-4 Single Sign-On Application pattern

Extended Single Sign-On Application pattern

This is an extended pattern where the security context is extended to include the back-end systems, to be able to use the Single Sign-On facility through a centralized security service.

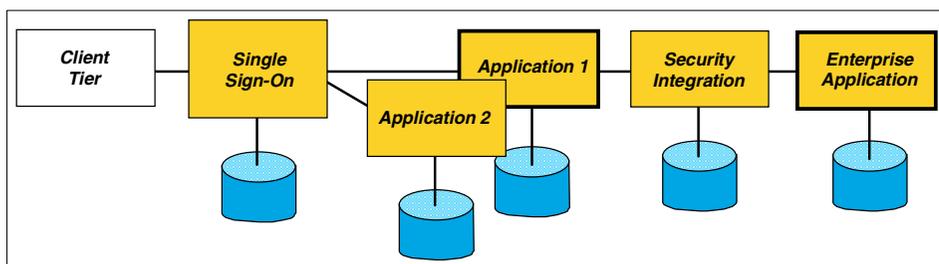


Figure 12-5 Extended Single Sign-On Application pattern

Together with the *Single Sign-On* tier and the *Security Integration* tier, this ensures the secure interaction with the back-end *Enterprise Application tier*.

Now we move to the next logical step in the solution design process: Runtime patterns.

12.3 Runtime patterns

Runtime patterns define functional nodes that underpin an Application pattern. Runtime patterns depict the solution architecture at a very high level, where networks and nodes are identified but no product or technology selection is made. Most patterns will consist of a core set of nodes, with the addition of one or more nodes unique to that pattern. To understand the Runtime patterns, you will need to review the following node definitions.

Nodes

The following nodes are defined in the upcoming Runtime patterns.

Client

The client node represents the user interface connecting to the server applications.

Public Key Infrastructure

Public Key Infrastructure (PKI) is a centralized security service for all of those clients who have certificates from the same Certificate Authority (CA). Certificates ensure the credibility of the communicating parties. PKI covers all the necessary infrastructure and services required to manage this type of security. This service can usually be found on the Internet, but intranets can also have their own PKI. PKI offers a technology which, if deployed properly, can provide greater security.

Protocol and domain firewalls

Firewalls provide services that can be used to control access from a less trusted network to a more trusted network. Firewalls also offer other services such as:

- ▶ Stateful packet inspection
- ▶ Logging
- ▶ Proxy support
- ▶ Address translation

The two firewall nodes provide increasing levels of protection.

Web server redirector

In order to separate the Web server node from the application server node, a so-called *Web server redirector node (redirector)* is introduced. The Web server redirector is used in conjunction with a Web server. The advantage of using a redirector is that you can move the application server behind the domain firewall into the secure network, where it is more protected than within the Demilitarized Zone (DMZ). Static pages and content can be served from the DMZ by this node.

Application server

The application server node provides the infrastructure for application logic. It might be part of a Web application server node. It is capable of running both presentation and business logic. When used with a Web server redirector, the application server node will run presentation and business logic. In other situations, it may be used for business logic only.

Web application server node

A Web application server node is an application server that includes an HTTP server (Web server) and is typically designed for access by HTTP clients and to host presentation and business logic.

The Web application server node is a functional extension of the informational (publishing-based) Web server node. The node provides robust services to allow users to communicate with shared applications and databases; in this way, it acts as an interface to business functions.

Directory services

The directory services node supplies information on the location, capabilities and various attributes (including user ID/password pairs and certificates) of resources and users known to the system. The node may supply information for various security services (authentication and authorization).

Security services

Security services perform actual security processing such as authentication and authorization.

The authentication, in most current designs, validates the access to the Web server part of the Web application server node, but it can also authenticate for access to other resources.

Security services can go further and provide authorization for applications, where user access to certain resources is checked.

This node may be combined with Directory services into one node called Directory and Security services.

Database

The database server node provides a persistent data storage and retrieval service in support of transactional interactions. The data stored is relevant to the specific business interaction.

Web Security proxy

The Security proxy stands between the user and the Web application. The role of this node is to capture the user request from the client side before the request reaches any application. The Security proxy also serves as a single entry point for Web applications, ensuring a higher level of security and making the network less vulnerable. It allows both presentation and business logic to be relocated to a more secure network zone behind the DMZ.

Enterprise application

Enterprise application stands for applications at the back-end. These applications can be application servers, legacy systems or massive enterprise applications running businesses. This node is important in that the application is separated from the middleware nodes.

12.3.1 Basic Runtime pattern

The basic Runtime pattern illustrated in Figure 12-6 shows the simplest Runtime pattern including the Web application server with the minimum security components.

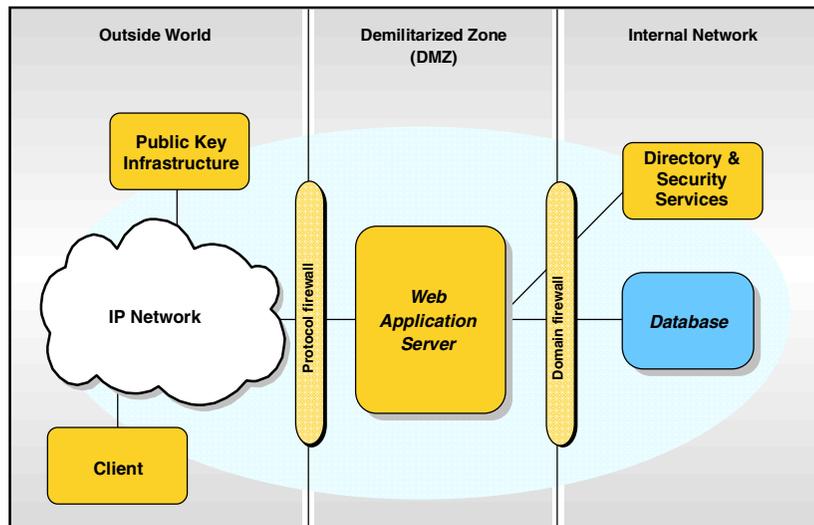


Figure 12-6 Basic Runtime pattern

The security is ensured by the two firewalls, which block network access from unauthorized networks and users.

Public Key Infrastructure provides certificate-based authentication.

Directory and Security services provide security functions for the Web application server to authenticate and authorize user access.

12.3.2 Runtime pattern variation

This variation of the Runtime pattern increases the security by separating the application server from the Web server. The application server is moved to the internal network.

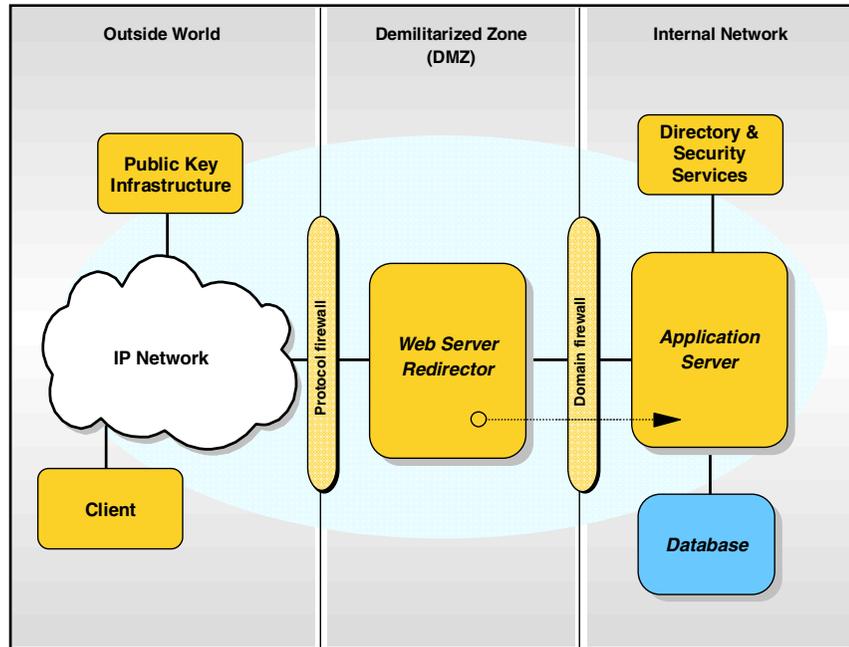


Figure 12-7 Runtime pattern variation

12.3.3 Single Sign-On Runtime patterns

The Single Sign-On runtime patterns bring new security components and aim to provide security services for a broader range of components.

Multiple application servers are running the e-business application(s); they can be the same application servers (homogeneous) or they can be different types from different vendors (heterogeneous). These application servers can connect to the core business applications (legacy systems).

The following four runtime pattern depict four different realizations of Single Sign-On with different features.

Enterprise applications in these patterns are represented with a node, but for back-end systems that are much more sophisticated, they can be placed in a separated network zone for higher performance and higher security reasons.

Single Sign-On runtime pattern for homogeneous application servers

The Runtime pattern illustrated in Figure 12-8 depicts the homogeneous Single Sign-On. The application servers are either the same server applications or are using exactly the same technique for Single Sign-On.

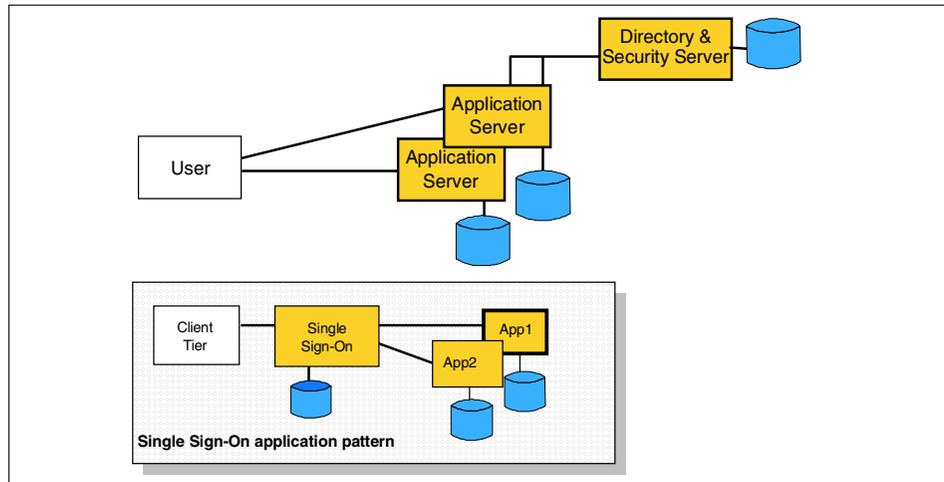


Figure 12-8 Single Sign-On runtime pattern for homogeneous application servers

Single Sign-On runtime pattern for heterogeneous application servers

As a variation of the previous Runtime pattern, Figure 12-9 depicts the case where the application servers are not running the same server application nor implementing the same Single Sign-On technique. Single Sign-On can only be provided by a Security Server external to the application server with a security proxy that intercepts requests to map/transform user credentials into the appropriate credential format for that application server.

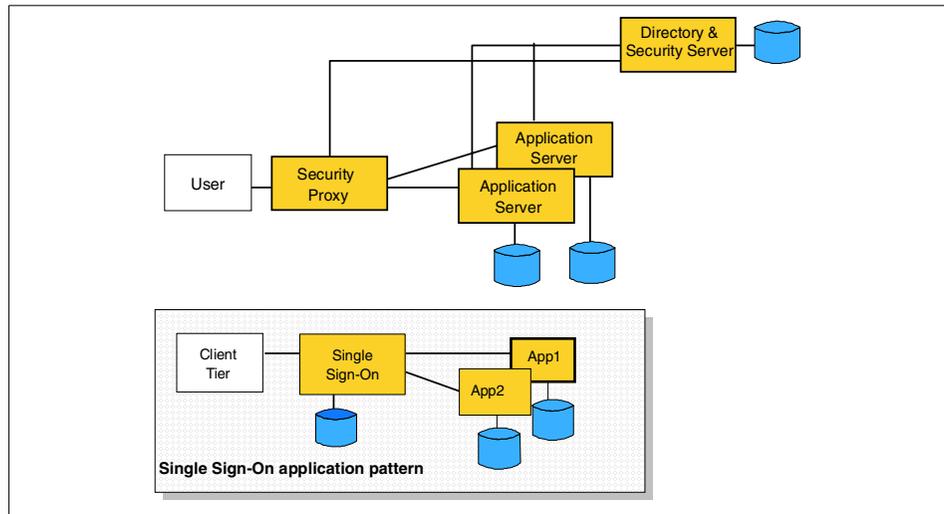


Figure 12-9 Single Sign-On runtime pattern for heterogeneous application servers

Example

WebSphere Commerce Suite is an example of an application that manages its own internal mechanism for authentication (form-based) and authorization (role-based). However, it is possible to establish trust relationships between WebSphere Commerce Suite and the larger enveloping domains of IBM WebSphere Application Server and Secureway Policy Director by sharing a common user registry or by configuring Commerce Suite to accept an LTPA token as proof of authentication. In this configuration, WebSphere Commerce Suite will accept the user as an authenticated user. Its authorization service will remain in effect and will determine permissions based on the user identity in the LTPA token.

Extended Single Sign-On runtime pattern for credential propagation

Extending the security context to back-end systems enables non-repudiation of transactions initiated by the user at the back-end. Credential propagation uses one of two approaches:

- ▶ Credential mapping, where the Web user identity is mapped to a user ID used to access the back-end system.
- ▶ Credential transformation, where the Web user identity is forwarded to the back-end system but is transformed into the format acceptable to that system.

Centralized security solutions can provide credential mapping services to solve this problem.

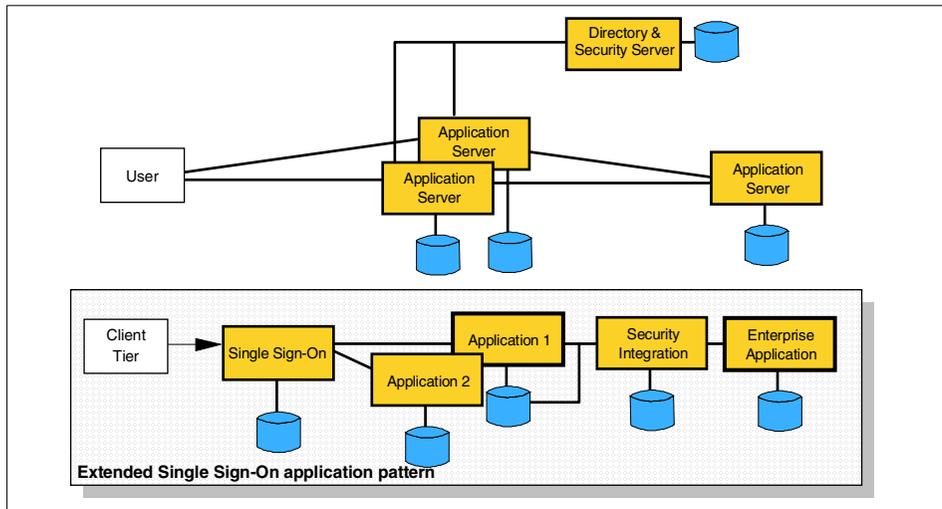


Figure 12-10 Extended Single Sign-On runtime pattern for credential propagation

For an example of the Extended Single Sign-On using credential propagation, refer to *Access Integration Pattern Using WebSphere Portal Server*, SG24-6267.

Extended Single Sign-On runtime pattern for central security service

Figure 12-11 depicts a Single Sign-On scenario with central security service.

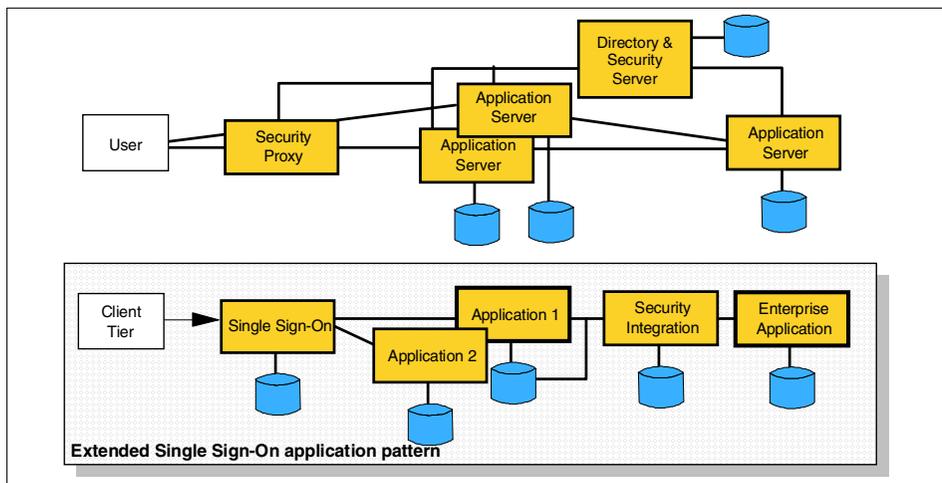


Figure 12-11 Extended Single Sign-On runtime pattern for central security service

This alternative allows for extending the security context to back-end systems and allows the same security service that controls the Web tier to control the back-end applications. The security server provides the role-based authorization for controlling back-end resources. No credential mapping or transformation is required. The security context is preserved all the way through to the back-end system.

The *security proxy* focuses on primary security functions such as authentication and authorization. Security is centralized in this node, which makes the system management easier. Security proxy provides a single access point for the e-business application, which makes the system less vulnerable.

12.4 Product mappings

The following implementations are examples, showing how security can be implemented in different scenarios. The approach is very similar to the runtime product mappings. The figures will show the security products applied to the Runtime patterns. These implementations are based on our experiences and the implementations used for this book.

12.4.1 Single Sign-On

The next two scenarios will introduce two different types of Single Sign-On architecture:

- ▶ Homogeneous Single Sign-On
- ▶ Heterogeneous Single Sign-On

Homogeneous Single Sign-On

With the basic Runtime pattern, the following Single Sign-On scenarios are available:

- ▶ Between two or more WebSphere Application Servers
- ▶ Between WebSphere Application Server and Lotus Domino

The scenario is depicted in Figure 12-12 on page 347; the application servers are placed in the DMZ.

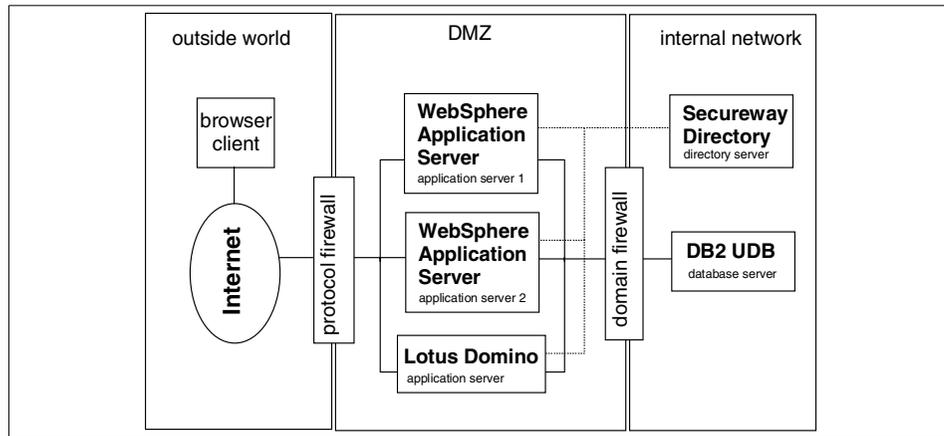


Figure 12-12 Homogeneous Single Sign-On

In this case, the application servers are using the same method to resolve user login for the session. WebSphere Application Server and Lotus Domino can use the same LTPA token to determine user credentials, as long as the application servers are in the same realm.

Despite the different products, this scenario is still homogeneous, because the products can share the login information without any additional code or component. Refer to Chapter 14, “Single Sign-On” on page 393 for an example of homogeneous Single Sign-On.

Heterogeneous Single Sign-On

Figure 12-13 on page 348 shows a scenario where different application servers and enterprise applications have a common login. It is a more usual case than that of the homogeneous SSO.

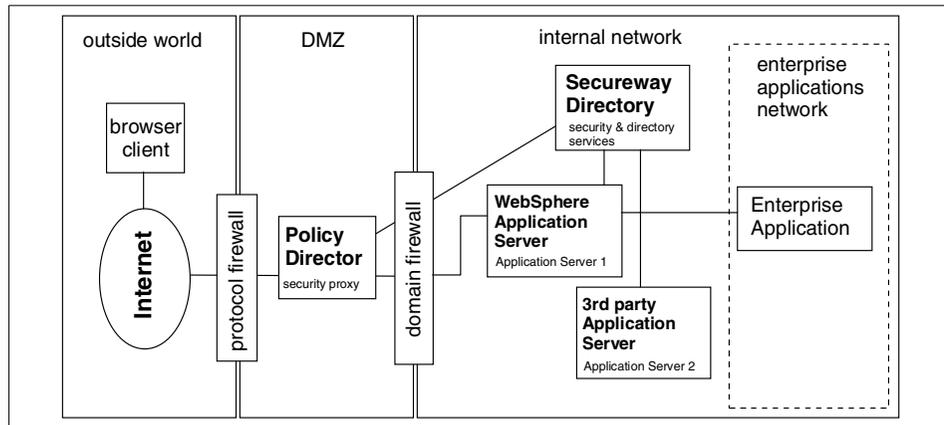


Figure 12-13 Heterogeneous Single Sign-On

The heterogeneous scenario was not tested in this book, and only a theoretical explanation of Single Sign-On will be documented here.

Heterogeneous Single Sign-On means that the products need additional code or components to achieve Single Sign-On.

Tivoli Policy Director provides a solution for heterogeneous systems called *Global Sign-On (GSO)*. The login information is stored in an LDAP directory, and the applications can check that information for themselves. Policy Director also provides APIs for applications to develop the code necessary to use GSO.

In a large scale system where security is centralized, Single Sign-On is a standard requirement; in certain cases, Global Sign-On can also be exercised.

Note: What is the difference between Single Sign-On and Global Sign-On from a technical point of view?

Single Sign-On works between systems which are using the same mechanism for user login. The systems can pass the login information between each other without any difficulties.

Global Sign-On works between systems where the user login mechanism is different on each implementation. In this case, the user login information is stored in a centralized directory, and each application has to access this directory to find out whether or not the user is logged in.

Refer to Chapter 13, “Policy Director” on page 353 for more details.

An issue to be aware of within a heterogeneous environment is that different vendor systems might use different credentials for users. One solution for this problem is *credential mapping*, where the security server maps the original user credentials to the application's request.

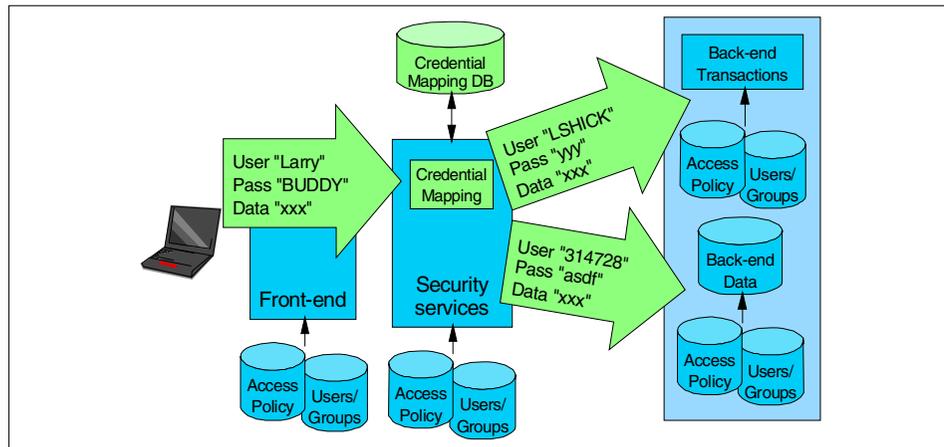


Figure 12-14 Credential mapping

In Figure 12-14, the user accesses the system from a client (for example a Web browser) through a front-end application. The front-end application has to connect to back-end applications. The same user has different credentials for different applications. Security services provide credential mapping to propagate the login information as it is required for the different applications.

For more details on Tivoli Policy Director, refer to the IBM Redbook *Enterprise Security Management with Tivoli*, SG24-5520; and *Tivoli Policy Director: Centrally Managing e-business Security*, SG24-6008.

12.4.2 Centralized security

There is a need for centralized security when systems grow large, not only to achieve easier management, but also to avoid security holes. It is very difficult to see the solution end-to-end and manage all the security elements separately. The following architecture realizes the centralized security on two different levels:

- ▶ Single entry point for the clients
- ▶ Centralized security services for the applications

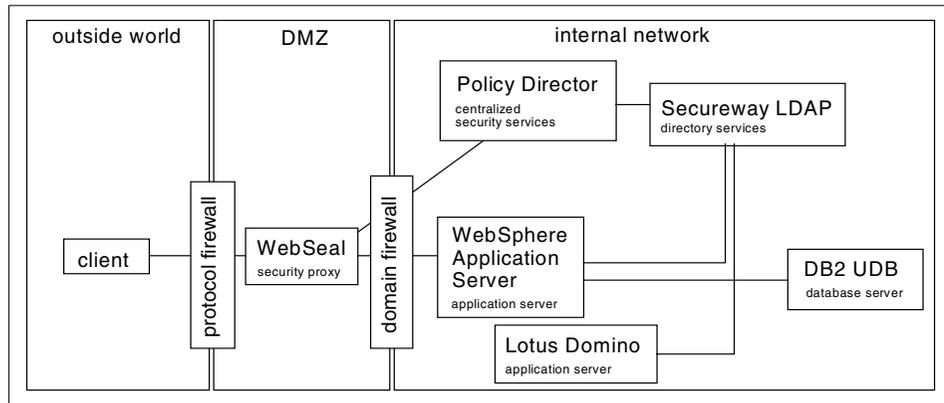


Figure 12-15 Centralized security

The node with WebSEAL provides a single access point to the system. Every client must go through this point to get access to the applications. WebSEAL is connected to the Policy Director, which provides the security services for the whole system.

In this example, only WebSphere Application Server and Lotus Domino are deployed, but in a general architecture, any other application server (third-party application servers) can be deployed and back-end applications are also available in a large environment.

SecureWay LDAP provides the directory services to the system, and both WebSphere Application Server and Lotus Domino Server use it for their user registry.

Note: WebSEAL is a component of the Tivoli Policy Director package.

12.5 More information

The most valuable source of information about patterns is the book *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, George Galambos, Srinivas Koushik, and Guru Vasudeva, ISBN 1931182027.

For more information about Access Integration security, refer to the following IBM Redbook *Applying the Patterns for e-business to Domino and WebSphere Scenarios*, SG24-6255.

For more information about Patterns for e-business, see the following Web site: <http://www-106.ibm.com/developerworks/patterns>, or refer to the following IBM Redbooks:

- ▶ *Applying the Patterns for e-business to Domino and WebSphere Scenarios*, SG24-6255
- ▶ *Self-Service Patterns using WebSphere Application Server V4.0*, SG24-6175
- ▶ *User-to-Business Pattern Using WebSphere Personalization Patterns for e-business Series*, SG24-6213
- ▶ *Self-Service applications using IBM WebSphere V4.0 and IBM MQSeries Integrator Patterns for e-business Series*, SG24-6160
- ▶ *e-commerce Patterns for Building B2C Web Sites, Using IBM WebSphere Commerce Suite V5.1*, SG24-6180
- ▶ *Access Integration Pattern using IBM WebSphere Portal Server*, SG24-6267
- ▶ *Mobile Applications with IBM WebSphere Everyplace Access Design and Development*, SG24-6259



Policy Director

In this section, we will look at security beyond what IBM WebSphere Application Server provides.

WebSphere's security features allow you to protect your e-business applications from unauthorized users. This is not all that is required for a secure end-to-end solution. Other security issues must be dealt with, such as:

- ▶ What other related systems and resources need to be protected?
- ▶ How can I integrate my Web application security with my existing application security infrastructure?
- ▶ How can WebSphere security interoperate with other applications in my enterprise?

We will look at each of these questions in turn. In “End-to-end security solutions” on page 354 we will consider the systems and resources, in addition to WebSphere, that are required for a fully secure solution. Then, in “Using Tivoli Policy Director” on page 355, we will see how WebSphere can be used with a specialized enterprise security product.

13.1 End-to-end security solutions

WebSphere security is not all that is required to protect your enterprise resources. It is only one piece in an end-to-end secure solution. There are several levels on which security breaches can take place and all of them need to be addressed in order to have a secure solution:

- ▶ Network level
- ▶ Operating System level
- ▶ Middleware level
- ▶ Application level
- ▶ Human level

Each of these levels is susceptible to different kinds of security breaches which must be protected against in the most effective way.

Networks are prone to many types of attack. Here are some examples of TCP attacks:

- ▶ Denial of service: the attacker sends thousands of request for connection to a server. The unresolved requests fill up the buffers in the TCP/IP stack, which stops new connections from being allowed to the server.
- ▶ Source address spoofing: you can never trust the IP address of incoming connections.
- ▶ Network sniffing: an intermediary listening on the network can see all the traffic.

TCP/IP stacks themselves can provide some protection nowadays. Firewalls, virtual private networks (VPNs) and third party encryption tools are all necessary tools to protect your networks and systems against many of these attacks.

Operating systems must be protected from unauthorized access. Hardening your file system and limiting the access to important files are both vital actions in implementing an overall secure solution.

Care needs to be taken with middleware application servers. You may have the tightest security on your operating system and your applications may only allow the right people to access the right information, but if you forget to secure the administration tools with passwords, then anyone could modify the server configuration. Care needs to be taken especially with Web-based administration tools.

Your applications need to be protected from the instance of one person getting access to another person's information. This could mean forcing Web users to identify themselves with a user name and password when they start using your Web site. Programmatic security, middleware functionality or third party security tools could all help in this case.

Consider also that all the security products and processes in the world have no effect if you cannot trust the people who look after them. This includes controlling physical access to the systems which contain your sensitive data and business logic.

All these concepts need to be considered when designing a secure solution. WebSphere provides security features for protecting the application server (middleware) and the applications it runs. Firewalls, system hardening and physical security all need to be considered as well.

13.2 Using Tivoli Policy Director

Tivoli Policy Director (PD) provides a powerful solution for network security management and Single Sign-On for Web servers and other application servers. Tivoli PD manages the authentication and authorization of servers and Web sites and provides an API interface so that you can extend these security services to nearly any other system in your enterprise. PD uses a centralized database to store its policy information; this allows new systems to be added without having to change your security infrastructure.

Securing production applications are vital to all organizations. Using WebSphere's own security features is sufficient in many situations, but if you want a consistent security infrastructure across all your middleware applications, then integrating WebSphere applications with PD becomes necessary.

13.2.1 Using Tivoli WebSEAL

WebSEAL is one of the components that ships with Tivoli Policy Director. It is an example of a Reverse Proxy Security Server and can protect the object space of itself and other Web servers in the enterprise.

WebSEAL can be integrated with WebSphere to provide the authentication services. WebSphere still provides the authorization services.

WebSEAL uses several elements to configure access control for a particular Web object space.

- ▶ **Junctions:** used to define a particular Web server's object space for the WebSEAL server.
- ▶ **Access control lists (ACL):** used to assign access permissions.
- ▶ **Groups:** a registry element that contains a number of users.
- ▶ **Users:** a registry entry that defines a particular user or system.

Figure 13-1 shows the relationship between each of these elements.

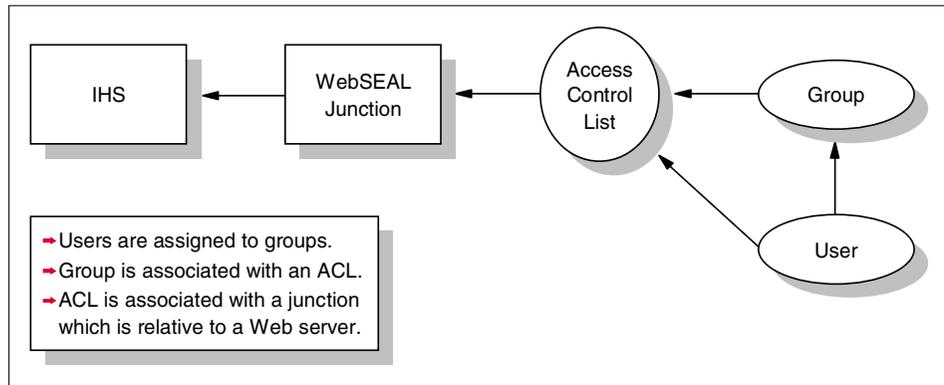


Figure 13-1 Relationship between junctions, ACLs, Groups and Users

Figure 13-2 shows the flow of a request from the browser to the application server.

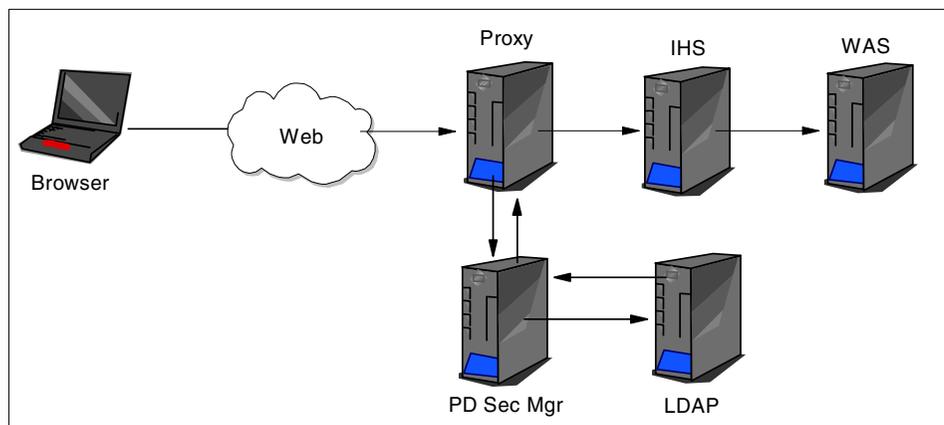


Figure 13-2 HTTP request flow from browser to application server

In the rest of this section, we will demonstrate several ways to configure WebSEAL to authenticate resources belonging to the Web server and to WebSphere.

The junctions that we will define in this section are:

- ▶ TCP junction: used to protect static resources belonging to the Web server.
- ▶ LTPA junction: used for LTPA authentication by WebSEAL.
- ▶ SSL junction: used for Web Trust Association between WebSEAL and WebSphere.

Figure 13-3 shows in more detail which requests are made and where the different components would exist within an enterprise.

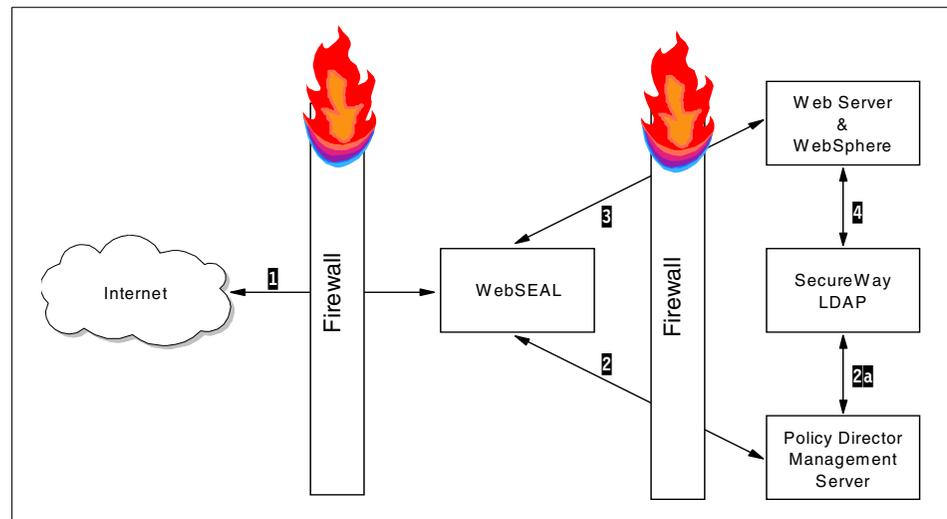


Figure 13-3 Typical infrastructure components with WebSphere and Policy Director

1. An HTTP request for a secured WebSphere resource is made, and WebSEAL receives it first.
2. WebSphere makes a request to Policy Director Management server to authenticate the user.
 - a. Policy Director authenticates the user against the LDAP registry and passes the result, success or failure, to WebSEAL.
3. WebSEAL forwards the request to WebSphere.
4. WebSphere determines if the user is authorized to access that particular resource by checking the security information in the deployment descriptors and checking group information in the LDAP registry.

Note: If you have enabled security on the IBM HTTP Server (see “How to secure HTTP basic authentication for IBM HTTP Server” on page 106), then make sure this is disabled before following the instructions in this section.

13.2.2 Using Tivoli Policy Director to protect static pages

In this section, we will describe how to configure Policy Director to protect the static resources that belong to the Web server, in our case the IBM HTTP Server. We will use basic authentication. In our environment, we installed all the software onto one machine.

First, install and configure:

- ▶ IBM HTTP Server V1.3.19

Then, from the Tivoli Policy Director Base product CD, install and configure:

- ▶ IBM Global Security Toolkit
- ▶ IBM SecureWay Directory Server V3.2.1
Install both the client and the server. If you do not have a version of DB2 installed, then SecureWay can install this at the same time.
- ▶ Tivoli Policy Director Runtime Environment
- ▶ Tivoli Policy Director Management Server
- ▶ Tivoli Policy Director Authorization Server

Finally, from the Tivoli Policy Director WebSEAL product CD, install and configure:

- ▶ Tivoli Policy Director WebSEAL

See the product installation guides for more information. Make sure all these services are started.

The key steps in securing the Web server resources with WebSEAL are:

- ▶ Creating a WebSEAL junction relative to the Web server.
- ▶ Creating a Group and associating some users with it.
- ▶ Creating an Access Control List (ACL) and associating a Group and access permissions with it.
- ▶ Associating the ACL with the junction.

Below are the step-by-step instructions you will need. Refer to the Tivoli Policy Directory Base Administration Guide for more details on the commands we use here.

1. First, we need to change the port that IHS is listening on.

Note: WebSEAL and IHS are both Web servers which, by default, listen for HTTP requests on the same port: port 80. Because we are running them both on the same machine, we need to change this listening port for one of the servers to avoid conflicts. We will change the port on IHS.

In practice, you will probably have WebSEAL and IHS running on separate machines. In this case, you may still want to change the port that IHS is listening on. Certainly, if you want to protect the Web server resources with WebSEAL, you need to make sure that traffic cannot access the Web server directly. One way this could be done is not to allow traffic for the port that IHS is listening on through the firewall.

Edit the IHS configuration file, C:\IBM HTTP Server\conf\httpd.conf. Look for the **Port** command and change the default value 80 to a new port number, for example **8080**. So the modified command looks like this:

```
# Port: The port the standalone listens to.  
Port 8080
```

Save the file. Now stop and start the IBM HTTP Server service.

2. Start the *pdadmin* tool. From a command prompt, type **pdadmin**.

```
C:\> pdadmin  
pdadmin>
```

Now log in with the security master's user name and password. Type **login** at the command prompt, then, when instructed, enter the user name and password.

```
pdadmin> login  
Enter User ID: sec_master  
Enter Password:  
pdadmin>
```

3. From the **pdadmin** command prompt, create some new users. Create at least two. The syntax is as follows:

```
pdadmin> user create [-no-password-policy] user name <distinguishedName>  
<commonName> <surName> password
```

The two we created were:

```
pdadmin> user create -no-password-policy peter "cn=Peter  
Kovari,c=US,o=webbank" peter peter peter  
pdadmin> user create -no-password-policy joanna "cn=Joanna  
Hodgson,c=US,o=webbank" joanna joanna joanna
```

This syntax created the following users:

Table 13-1 Users

User Name	Password
peter	peter
joanna	joanna

These users are created in the SecureWay LDAP server. You can use the Directory Management Tool to see them.

Note: There needs to be a parent entry listed in the LDAP directory that you can add users to. That is, a DN of `c=US,o=webbank` must exist in the user registry before you can add users.

4. We now need to make these users' accounts active so that they are shown to be valid users when they are given access to resources. From the `pdadmin` command prompt, use the following syntax:

```
pdadmin> user modify <userName> account-valid yes
```

In our case, the commands were:

```
pdadmin> user modify peter account-valid yes  
pdadmin> user modify joanna account-valid yes
```

5. Verify that you can log onto the WebSEAL Web server. Point your browser to `https://<WebSEALfullHostname>`.

The first thing you will probably see is a security information box warning you that the certificate that the Web server has presented to you is not known by your browser. This certificate is needed to establish the secured, encrypted connection between the browser and the Web server. These information boxes are different in different browsers. Here we show the details for Internet Explorer.

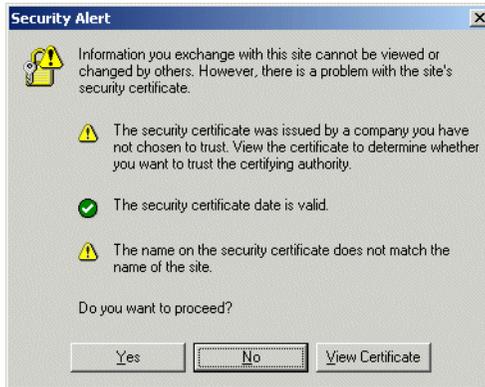


Figure 13-4 Security information warning from Internet Explorer

Choose to proceed (in the case of Internet Explorer, select **Yes**). Once the security warnings have been dealt with, you will be presented with the basic authentication challenge to log in to your Web site.



Figure 13-5 Basic login prompt for WebSEAL

Enter a valid user name and password as created in steps 3 and 4 above and click **OK**. A successful login will produce the page shown in Figure 13-6 on page 362.

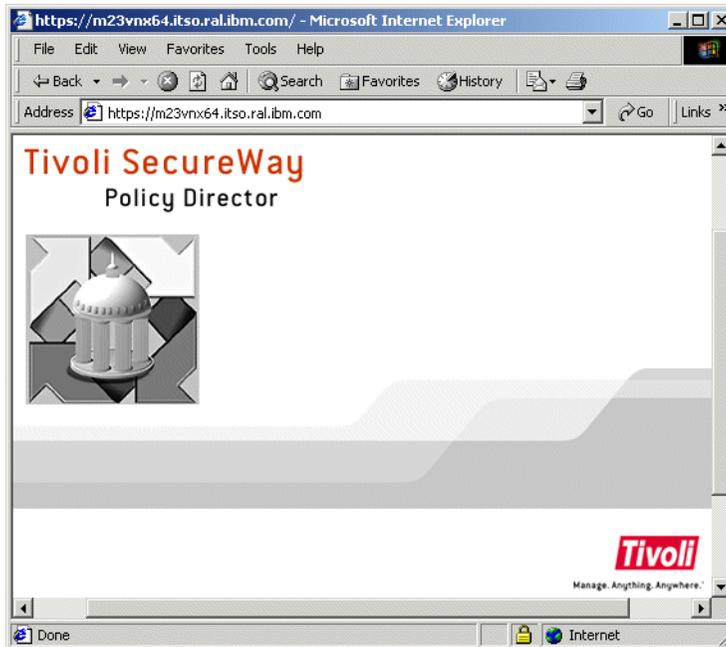


Figure 13-6 Successful login to your WebSEAL Web site

6. Check the name of your WebSEAL server, as defined in Policy Director. It is of the format `webseald-<shortHostname>`. To list all the servers that are defined to PD, from a `pdadmin` command prompt, enter:

```
pdadmin> server list
```

Our server used for this example is `webseald-m23vnx64`.

7. Now, create a WebSEAL junction to associate WebSEAL with your Web server. We will create a TCP junction rather than an SSL junction, that is, the communication between WebSEAL and the Web server will not be encrypted. The syntax is:

```
pdadmin> server task <WebSEALServer> create -t tcp -h
<WebServerFullHostname> -p <WebServerPort> </junctionName>
```

In our case, the command we used was:

```
pdadmin> server task webseald-m23vnx64 create -t tcp -h
m23vnx64.itso.ral.ibm.com -p 8080 /httpserver
```

This created a junction called `/httpserver` to our Web server, IHS, listening on port 8080. The full name of our junction in the object space is `/WebSEAL/m23vnx64/httpserver`.

8. Create a group called *employees*. We will add just one of our valid users to this group so that we can demonstrate that the access controls are working.

The syntax for creating a group is:

```
pdadmin> group create <groupName> <groupDistinguishedName>
<groupCommonName> <groupContainer>
```

In our example, we used:

```
pdadmin> group create employees cn=employees,c=US,o=webbank employees
WEBBANK
```

If you use the Directory Management Tool for the LDAP server to view your user registry, you will now see this new group entry.

9. Now we will add one of our two valid users to this group. The syntax for this process is:

```
pdadmin> group modify <groupName> add <userName>
```

In our example, we used:

```
pdadmin> group modify employees add joanna
```

Again, you can check that this user has been added to the group using the LDAP tool.

10. We now need to create an access control list (ACL) to associate with the WebSEAL junction we created earlier. The syntax to create an ACL is:

```
pdadmin> acl create <aclName>
```

For our example, we used:

```
pdadmin> acl create webbank-acl
```

11. For this ACL to be of use, we need to define some users and/or groups to it. For this example, we will add the *employees* group which contains a single user. The syntax for this is:

```
pdadmin> acl modify <aclName> set group <groupName> <permissions>
```

In our example, we used:

```
pdadmin> acl modify webbank-acl set group employees Trx
```

The permissions T, r and x refer to:

Table 13-2 Permissions descriptions

Permission	Description
T	Traverse sub-directories
r	Read permission
x	Execute permission

12. Finally we must associate this ACL to the junction. The syntax is:

```
pdadmin> acl attach <fullJunctionName> <aclName>
pdadmin> acl attach /WebSEAL/m23vnx64/httpserver webbank-acl
```

This will enable the user *joanna* to read, execute and traverse objects on the Web server. The user *peter* should not be able to access any of the Web server resources.

13. Now test that the result is what you expected. When you point your browser to `https://<WebSEALfullHostname>/<junctionName>/index.html`, this refers to the IHS home page. If you try to log in with the user name *peter*, you should get an authentication failure. Logging in as *joanna* should present you with the IHS page.

In our example, the URL we used was:

```
https://m23vnx64.itso.ral.ibm.com/httpserver/index.html.
```

If you are warned about the SSL certificate, choose to proceed. When prompted to enter a user name and password, first enter the user that should not work, in our case, *peter*. You should get an error page like the one in Figure 13-7.



Figure 13-7 Error page received when using an invalid user name

Now restart your browser and try again. This time use a valid user name, in our case *joanna*. You should see the IHS home page as shown in Figure 13-8.



Figure 13-8 Successful login by a valid user

13.2.3 Using Tivoli Policy Director to protect WebSphere URIs

The instructions in this section do not involve any WebSphere security. Here we will use Policy Director to secure servlets and JSPs which belong to WebSphere. Policy Director cannot secure EJB resources. It also cannot be used to secure individual methods, only the URI. If this is sufficient for your requirements, then using just Policy Director would be sufficient in some circumstances.

For this section, we assume that the instructions in the previous section, “Using Tivoli Policy Director to protect static pages” on page 358, have also been completed. This is because the WebSphere resources in this case are just an extension of what we have already done. By protecting the Web server space, we have automatically protected anything that is accessed through it.

First, install and configure WebSphere Application Server V4. Make sure you install the Samples, as we will use them in this section.

1. From the WebSphere AdminClient, start the Default Server. Verify that the setup was successful by using *Snoop* servlet, accessing it through the embedded HTTP server, for example,
`http://m23vnx64.itso.ral.ibm.com:9080/servlet/snoop.`



Figure 13-9 Successful snoop servlet request

- Now we need to change the WebSphere configuration to accept requests coming from the IHS server. By default, WebSphere accepts traffic from port 80, so we need to change it for traffic coming from port 8080 (the port we changed it to in "Using Tivoli Policy Director to protect static pages" on page 358).

From the WebSphere Administration Console, expand the Domain and select **Virtual Hosts**. In the right-hand pane, make sure that **default_host** is selected. Next to Aliases, select **Add**. A new, blank line will appear in the list. Enter *:8080, then click **Apply**.

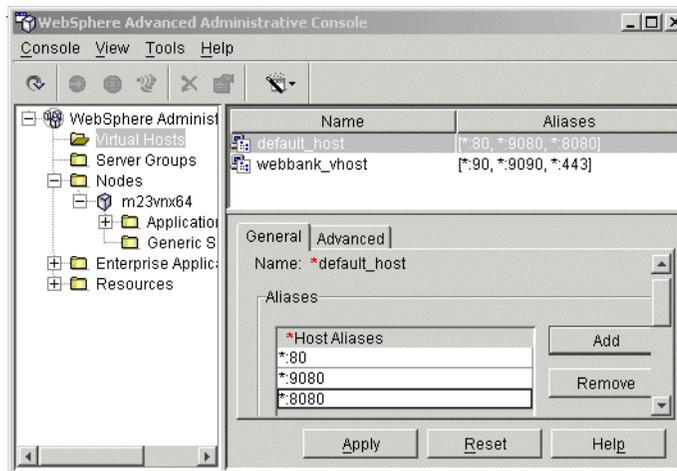


Figure 13-10 Virtual Host alias list

3. Update the Web server plug-in configuration file, `plug-in-cfg.xml`, to reflect that WebSphere will accept traffic from this port. To do this, expand **Nodes** and select your WebSphere node. In our case, the node is called `m23vnx64`. Right-click your node and select **Regen Webserver Plugin**.
4. Verify that IHS can communicate with WebSphere on this port by pointing your browser to `http://<fullHostname:8080>/servlet/snoop`. In our case, that is `http://m23vnx64.itso.ral.ibm.com:8080/servlet/snoop`.
5. Now access Snoop through Policy Director, using `https://<fullHostname>/httpserver/servlet/snoop`, and you should be prompted for a user name and a password as before. Enter a valid user, in our case, `joanna`, and you should be able to successfully access the Snoop servlet.
6. Try out some other applications from the WebSphere samples. For example, the HitCount application which you can use with EJBs, through `https://m23vnx64.itso.ral.ibm.com/httpserver/webapp/examples/HitCount`



Figure 13-11 HitCount web application: EJB model

It is possible to change the permissions on individual objects in the object space. For example, if you only want some people to access the Snoop servlet, you can apply a different access control list to just this object to restrict the users who can access it. The users you want to have access to one particular object must have access to the parent object.

If you use the Snoop servlet, then the users you want to access this must also have access to the /httpserver junction object space that we created earlier.

The following instructions extend what we have done so far and allow only user *peter* to access Snoop. User *joanna* is no longer allowed access.

1. First we need to add the user name *peter* to the group associated with the access control list for our junction. The group is called *employees*.

```
pdadmin> group modify employees add peter
```

User *peter* will now be able to access everything that user *joanna* can access.

2. Create a new group to use with the access control list for the Snoop servlet.

```
pdadmin> group create managers cn=managers,c=US,o=webbank managers
```

Now add user **peter** to this group.

```
pdadmin> group modify managers add peter
```

3. Create a new ACL. We will use this to control access to the Snoop servlet.

```
pdadmin> acl create webbank-mgr-acl
```

Now associate the group *managers* with this new ACL. Grant the users permissions Trx as before.

```
pdadmin> acl modify webbank-mgr-acl set group managers
```

4. Now attach this ACL to the Snoop servlet object:

```
pdadmin> acl attach /WebSEAL/m23vnx64/httpserver/servlet/snoop  
webbank-mgr-acl
```

This will allow user *peter*, but not user *joanna*, to access the Snoop servlet. The user *joanna* still has access to the other resources in the /httpserver object space.

To test this, point your browser to `https://<fullHostname>/servlet/snoop`. Try logging on as user *joanna*. You should get a Forbidden error as in Figure 13-7 on page 364. Try again with user *peter* and you should be successfully authenticated to access the Snoop servlet, as shown in Figure 13-9 on page 366.

13.2.4 Policy Director LTPA authentication

With Tivoli Policy Director 3.8, it is now possible for WebSEAL to generate and understand WebSphere LTPA tokens. This means that WebSphere will receive the request along with a valid LTPA token, so that WebSphere considers the user authenticated.

This method of authentication does not require WebSphere to be configured in any special way to allow Policy Director to provide this service, as with Web Trust Association (see Section 13.2.5, on page 381).

WebSphere and Policy Director do need to share the same LTPA keys. This is because LTPA tokens are encrypted; in order for WebSphere to verify that it has received a valid LTPA token, it must be able to decrypt it. Exporting the keys from WebSphere and importing them into Policy Director is a manual process.

This method of integration between Policy Director 3.8 and WebSphere is very useful, especially if the version of WebSphere you are using does not support Web Trust Association.

Note: With previous versions of WebSphere, you needed to enable SSO for the Policy Director LTPA authentication to work correctly. This meant that WebSphere would send a cookie containing an LTPA token back to the browser with the HTTP response. This cookie was not used or required for Policy Director LTPA to work. However, because this cookie, containing the LTPA token, was sent back to the browser, there was exposure. The LTPA keys could be cracked in an off-line attack. This meant that it was important to periodically regenerate the LTPA keys within WebSphere and redistribute them to Policy Director.

With WebSphere V4 SSO it is no longer required for PD LTPA authentication to work. Even if it is enabled, the LTPA token is no longer sent to the browser, eliminating this exposure.

For our purposes, we will use this method to authenticate access to our Webbank application. The following steps will be required:

- ▶ Configure WebSphere to use the same LDAP repository as Policy Director.
- ▶ Enable security for the Webbank application.
- ▶ Export the LTPA keys from WebSphere.
- ▶ Create a WebSEAL junction to generate the LTPA tokens using the LTPA keys.

For this example, we used a configuration slightly different from the previous one. Here we have a two machine configuration: one with SecureWay LDAP, Policy Director and WebSEAL, and a second with IBM HTTP Server and WebSphere (see Figure 13-12 on page 370).

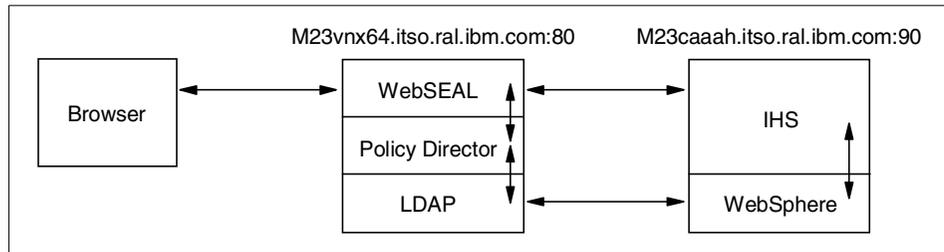


Figure 13-12 Two box configuration used in this LTPA scenario

1. Create a WebSphere administration ID to access the same LDAP that Policy Director is using:

```
pdadmin> user create wasadmin "cn=WAS Admin,c=US,o=webbank" wasadmin
wasadmin passw0rd
pdadmin> user modify wasadmin account-valid yes
```

Note: When you installed Policy Director, it made changes to the default SecureWay LDAP server configuration. In particular, by default, a user does not have the permissions to read or search the registry. These permissions are required by the WebSphere administration user name. The best way to do this is to add an LDAP ACL to this user which provides the right level of access.

2. Modify the LDAP ACL for this user. From the SecureWay Directory Management Tool, click **Add Server** in the bottom right-hand corner of the window.

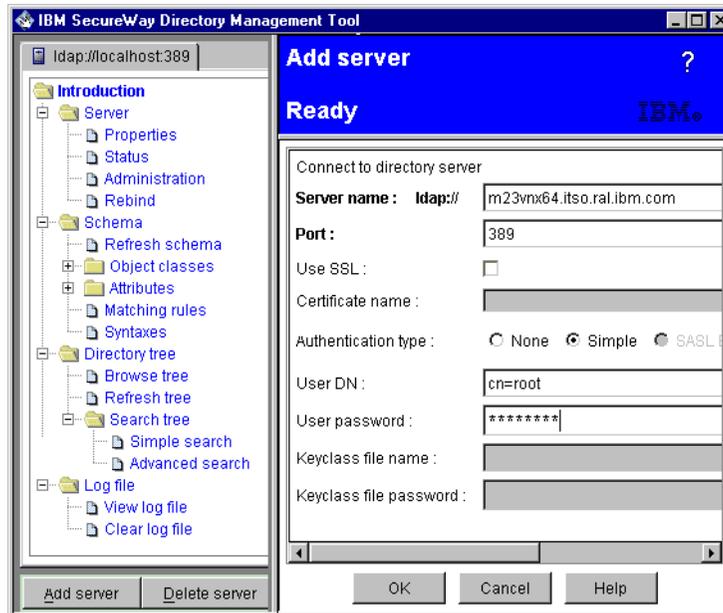


Figure 13-13 Adding a new server entry panel

3. In the right-hand panel, enter:
 - The server name: ldap:// <fullHostname>
 - The authentication type: Simple
 - The user DN: cn=root
 - The user password: <rootPassword>

Click **OK** to finish. You will see a second tree structure in the left-hand panel for this server definition.

4. Click **Browse tree** to see the list of defined users and groups.

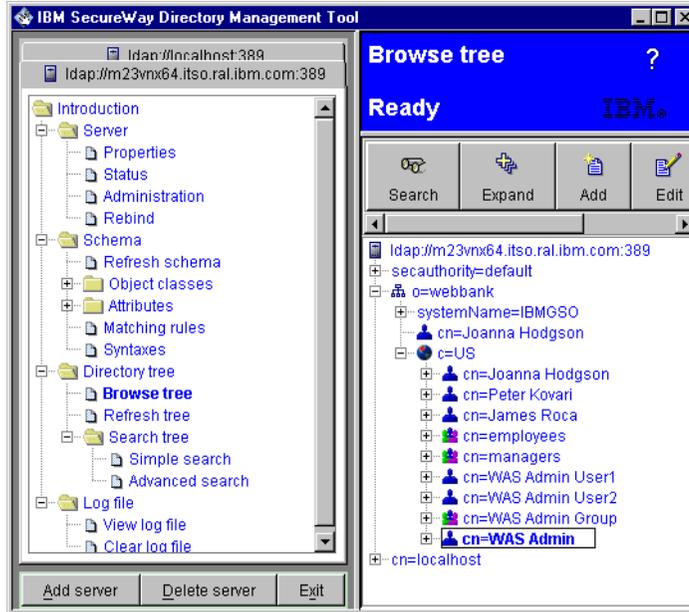


Figure 13-14 LDAP server registry tree structure

5. Double-click your new user, **cn=WAS Admin**, to bring up the Edit User Properties window.

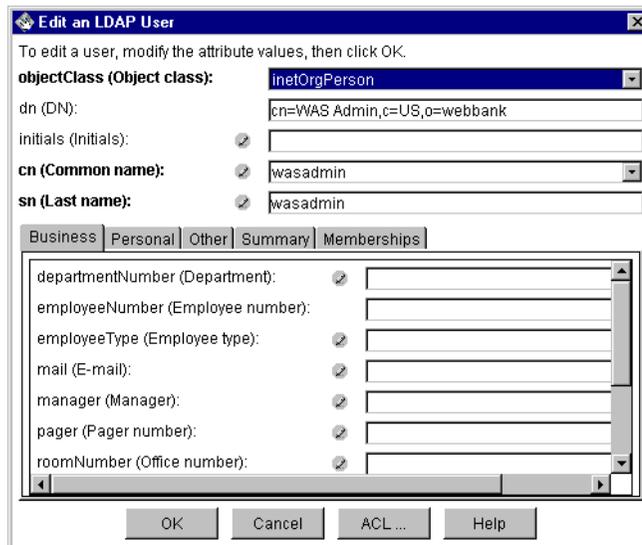


Figure 13-15 Edit user properties box

- Click the **ACL** button to bring up the window allowing you to edit the user's access privileges. This is where we are going to enable the user to read and search the user registry so that this user's information can be referenced for the WebSphere administration user name.

Under the ACLs tab, provide the following information:

- Distinguished name (DN): CN=ACCESSID
- Type: accessid

Click **Add**.

- Now change the Security classes so that Read and Search permissions are granted for each class: *normal*, *sensitive* and *critical*. A completed ACL panel should look like that shown in Figure 13-16.

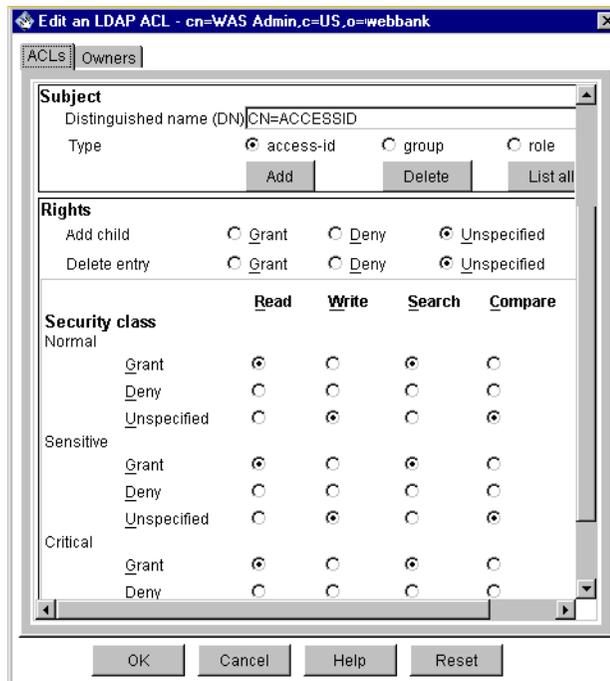


Figure 13-16 Access control for wasadmin user

- Now select the **Owners** tab. For the Distinguished name, enter cn=WAS Admin,c=US,o=webbank. For the type, select **access-id**. Then click **Add** to save the change.

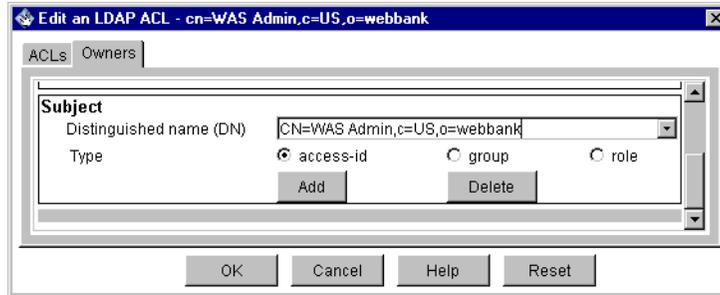


Figure 13-17 Assigning the access permissions to the wasadmin user

9. Click **OK** to close the ACL entry box.

10. Click **OK** again to close the Edit User Properties window.

You now have a user name that can be used for the WebSphere administration ID.

11. Here we assume that you already have the Webbank application installed on your node. If you have enabled security for the application before you installed Policy Director, then you may need to make some changes to your global security settings.

From the WebSphere administration console, start the Security Center by clicking **Console -> Security Center**. Under the General tab, make sure that the Enable Security checkbox is selected. Now select the **Authentication** tab. Enter the following information:

Table 13-3 Authentication settings

Field	Value
Authentication Mechanism	Lightweight Third Party Authentication (LTPA)
Enable Single Sign On (SSO)	Yes
Registry	LDAP
Security Server ID	The user name you created in step 1
Security Server Password	The password for the user name
Host	IP address of your LDAP server
Directory Type	SecureWay
Port	389
Base Distinguished Name	o=webbank

Field	Value
Bind Distinguished Name	Full DN of user name created in step 1
Bind Password	Password

The values that we used are shown in the figure below.

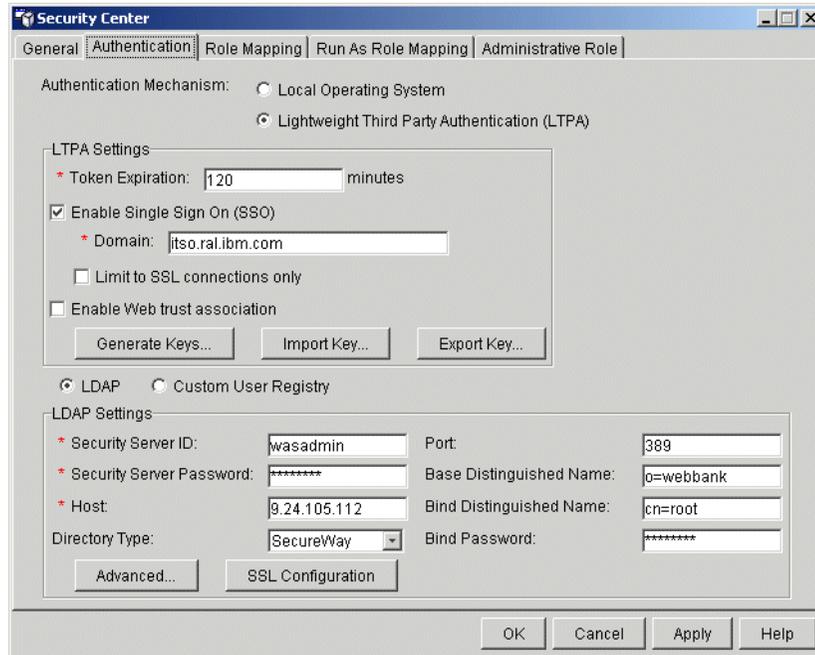


Figure 13-18 Global authentication settings

12. For the example in this chapter, we will be using groups and users defined from Policy Director, so we need to change the search filter for groups.

Note: The default SecureWay configuration for WebSphere will search for users in the LDAP registry using the object class *ePerson* and search for groups using the object classes *groupOfName* and *groupOfUniqueNames*. This is enough for users and groups created outside of Policy Director (PD), but PD creates groups based on the object class *accessGroup*. This means that the group filter for the SecureWay configuration needs to be modified if you are going to use groups that you create using PD.

Change the Directory Type to **Custom**. Click the **Advanced** button in the LTPA Settings section of the Security Center. You will be presented with the LDAP Advanced Properties pop-up window. Change the group entries to the following values:

Table 13-4 Group filter values

Field	Value
Group Filter	(&(cn=%v)(objectclass=accessGroup))
Group Member ID Map	accessGroup:member

So the completed panel will look like the one shown in Figure 13-19.

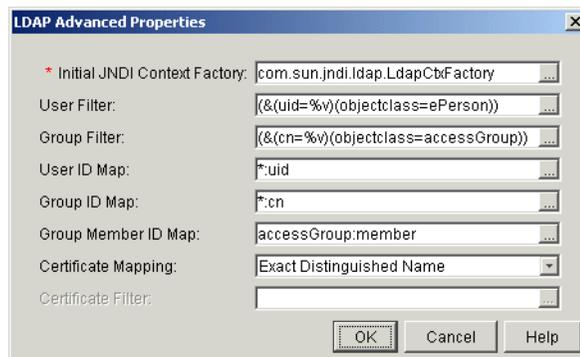


Figure 13-19 Modified search filter for SecureWay

Click **OK** to save the changes.

13. Click **Apply** in the Authentication pane to commit the changes.

14. Generate the LTPA keys by clicking the **Generate keys** button. You will be prompted with the LTPA Password entry window.

Note: If you do not do this now you, will be prompted when you click **OK**. However, because we want to export the keys before we finish, we need to have generated the keys first.

Enter a password and confirm it, then click **OK**. Remember this password, as it will be required to import the keys into Policy Director.

The LTPA keys are stored in the Admin repository. There are three keys that have been created:

- Private key: used to sign the LTPA token.
- Public key: used to verify the signature of the LTPA token.
- Shared key: a triple DES key used to encrypt and decrypt the LTPA token.

15. Export these keys to a file. From the Authentication pane of the Security Center, click the **Export key...** button. You will be presented with an **Export to File...** pop-up window.

Choose a location and name for your file, for example, c:\LTAPKeys.txt. Click **Save**. An ASCII file is created, containing the key information. An example of this file is shown in Example 13-1.

Example 13-1 LTPA exported key file

```
#
#Thu Nov 01 17:13:09 EST 2001
com.ibm.websphere.CreationDate=Thu Nov 01 17\:13\:09 EST 2001
com.ibm.websphere.ltpa.PublicKey=ALTMsay9hEW348PPww1HLzWYMYxsGaL0uHmtaPdZfZnXxq
NvLu+Fgd/E4uptPa85/RP5g9DgveQMxRfAr2GFWzQkFuA1w6RgELQa89eFYG7bzb+trAB1nbEukv6H2
02g0WduSLQ8EvITYRTXNr6gy+SsCRPAXSDWxW1b9d1MvTx1AQAB
com.ibm.websphere.ltpa.version=1.0
com.ibm.websphere.ltpa.Realm=m23vnx64.itso.ra1.ibm.com\:389
com.ibm.websphere.ltpa.PrivateKey=Hciu1KQVYqxh06QnNtvcCTdh4s0iyYko8RGVAHhRMQ95a
CeZAqGpHhMiiq9w3Edywh9obJD3bQTKyTc0Pz1Zxep+V2RF7cWf6jPXQY6c1Epc28gCZsxvoUmZKtFD
D/fvpmRhgF23osCOHT6DUGJkHVTZ2K7St44nsLfqtgnzqyPn9G31AA8JKbtstdKmP0xVy5auJ2oeg4J
AkkXskqaAf6YSBfMFupYS0aRFPqbu4YvKtAfCi1W9DUaA+XxirVrkqnL+8TQQLSvCKODC5UbNEqItcN
ZNcSkDhua1cC9dYgRYWeXRL1GKX/1d28b8r/LpqJ6CAG2d1tUHUdFynpb0r0m5nYxG/0sAo8a0BAhv1
JQ\=
com.ibm.websphere.CreationHost=M23CAAAH
com.ibm.websphere.ltpa.3DESKey=i/E201fUPL6tuWoS+LqPNfuDUR202eLPhbJvTv0VRrg\=
```

16. Now that you have finished configuring security, click **OK** to finish.

17. For the changes to take effect, you need to restart the WebSphere Admin Server. Right-click the node and select **Restart**. You will be prompted for confirmation, because this action will cause the administration console to close. Select **Yes**.

18. Table 13-5 on page 378 contains the users, groups and ACLs that we used for this example. Using the **pdadmin** command line tool, create similar entries. Make sure that all of the users have the account-valid flag set to Yes.

Table 13-5 List of users and groups

Group	Username	ACL	Permissions
employees	joanna, peter, james	webbank-emp-acl	Trx
managers	peter, james	webbank-mgr-acl	Trx
customers	victoria	webbank-cust-acl	Trx
(no group)	marko		

19. Now we will modify the Webbank role mappings to assign appropriate groups to the security roles for our application. Start the WebSphere Administration Console, select the **Webbank** enterprise application and go the User/Role Mappings tab.
20. Select the **Employee** role, then click **Select**. You will be presented with the **Select Users/Groups - Employee** entry window.
21. Deselect **All Authenticated Users** if it is selected, and check **Select users/groups** instead.
22. In the Search field, enter an asterisk (*) and click **Search**. This will list all the available users and groups in your user registry. Select the **employees** group (cn=employees) and click **Add**.

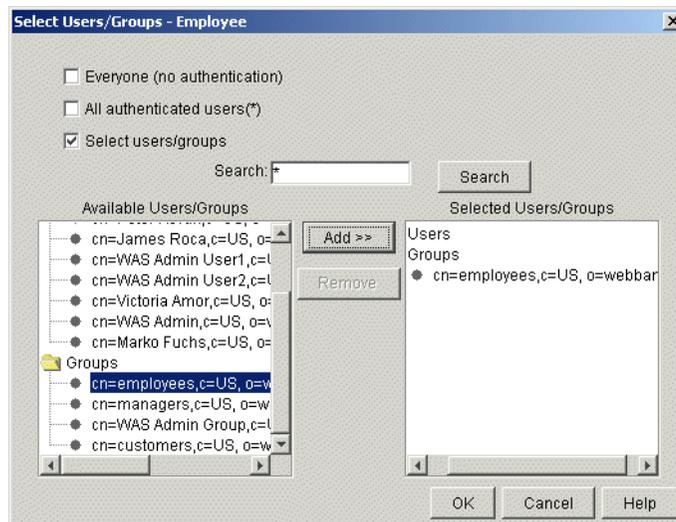


Figure 13-20 Selecting a specific LDAP group to map to a WebSphere security role

23. Click **OK** to save the change. Repeat this for the Manager role, assigning the **managers** LDAP group to it.

24. From the User/Role Mappings tab, click **Apply** for the changes to be committed. If you do not do this, your changes will be lost.

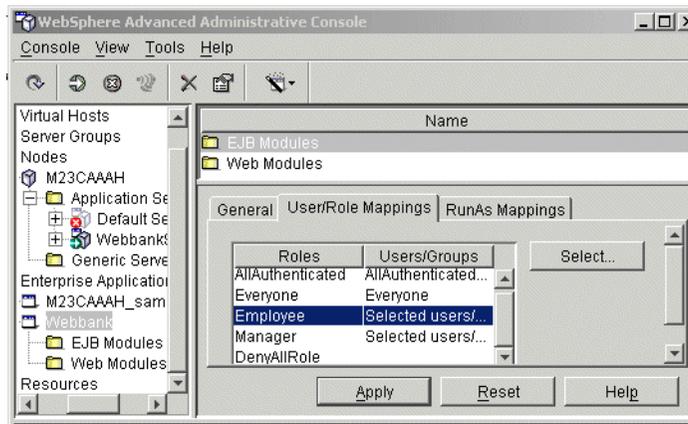


Figure 13-21 Completed User/Role mapping

25. Stop and start the Webbank application.
26. Verify that you can access the Webbank through IHS. Out of interest, you can view the LTPA token that is sent back to the browser as a cookie. In your browser settings, turn on cookie prompting. For Internet Explorer, select **Tools -> Internet Options** and choose the **Security** tab, then click the **Custom level...** button. Scroll down to **Cookies** and set *Allow per-session cookies (not stored)* to **Prompt**. Click **OK** and, when asked if you are sure you want to make this change, click **Yes**. Click **OK** in the Internet Options window.

Point your browser to `http://<fullHostname:90>/webbank`. You will be told that a cookie is going to be set. The browser will then be redirected to the login form. Accept this cookie. When prompted, enter an **Employee** or **Manager** user name. You will be prompted again with the same message announcing a cookie. This time, view the information in the cookie; in Internet Explorer, click the **More Info** button. You will see the LTPA token as a cookie.

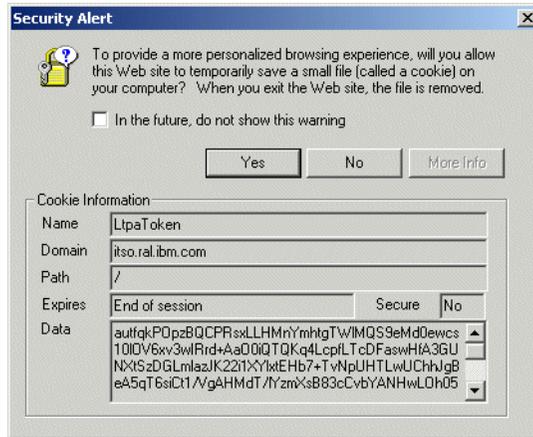


Figure 13-22 LTPA cookie

Click **Yes** to accept the cookie; you will see the Webbank home page. If you enter a user name that does not correspond to one of these roles, you will get a You are not authorized to view this page error message.

27. Now we will create the LTPA junction in WebSEAL so that Policy Director can perform the authentication for WebSphere. The syntax is:

```
pdadmin> server task <WebSEALServer> create -t tcp -h <IHSHostName> -p
<IHSPort> -j -i -A -F <LTPAKeyFile> -Z <LTPAPassword> </junctionName>
```

where

- i supports case insensitive URLs
- j supports WebSphere dynamically created URLs
- A supports LTPA cookies
- F is the LPTA key file
- Z is the LTPA password

The command we used in our environment was:

```
pdadmin> server task webseald-m23vnx64 create -t tcp -h
m23caaah.itso.ral.ibm.com -p 90 -j -i -A -F c:\LTPAKeys.txt -Z secret
/was1tpa
```

28. Policy Director can now authenticate the users for the Webbank application.

Verify this by pointing your browser to:

<https://<WebSEALHostname>/was1tpa/webbank>. If you enter a user name from the Employee or Manager role, you should be sent to the Webbank home page; if not, you should be denied access with the message You are not authorized to view this page.

If you still have the cookie prompt on, take a look at the contents of the cookie sent to your browser this time. You will see that it is a special WebSEAL cookie for the junction.

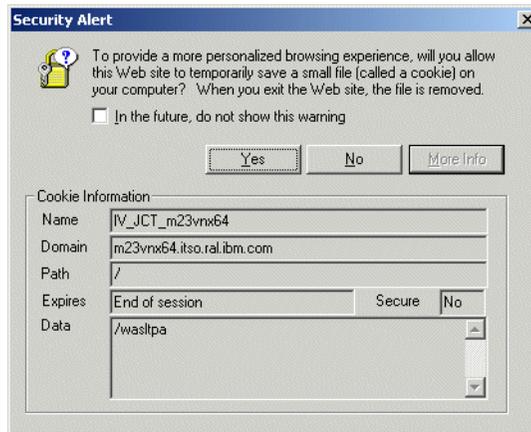


Figure 13-23 WebSEAL cookie for the LTPA junction

Note: Notice that even though SSO is enabled in WebSphere global security, the LTPA token cookie is not sent to the browser as it was when WebSphere was authenticating the user.

13.2.5 Web Trust Association

WebSphere has the ability to plug in a third party authentication product. The third party product must be a Reverse Proxy Security Server (RPSS); WebSphere communicates with this server through a plug-in called a Trust Association Interceptor.

If you want to use an RPSS to authenticate WebSphere applications, you must implement the `com.ibm.websphere.security.TrustAssociationInterceptor` interface. This interface defines three methods:

- ▶ `public boolean isTargetInterceptor(HttpServletRequest req)` throws a Web Trust Association exception.
- ▶ `public void validateEstablishedTrust(HttpServletRequest req)` throws a Web Trust Association exception.
- ▶ `public String getAuthenticatedUsername(HttpServletRequest req)` throws a Web Trust Association exception.

The plug-in is responsible for establishing trust between WebSphere and the RPSS. This is possible because the RPSS sends WebSphere a modified HTTP request, having substituted its credentials (user name and password) in the header. The plug-in also extracts this client's user name from a special HTTP header and passes it to the WebSphere security server to be used for authorization.

For WebSEAL, there is an implementation of the *TrustAssociationInterceptor* already provided with WebSphere V4. The following picture shows the typical flow of an HTTP request for a secured WebSphere resource authenticated by WebSEAL through a Web Trust Association.

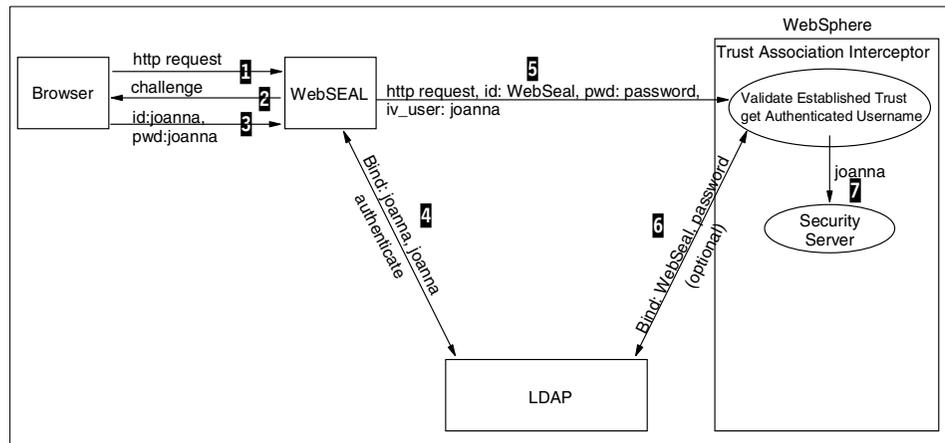


Figure 13-24 Web Trust Association authentication flow

1. The browser makes a request for a secured WebSphere resource.
2. WebSEAL sends back a challenge, either a HTTP Basic authentication or a form-based challenge.
3. User name and password are supplied.
4. WebSEAL authenticates the user against LDAP.
5. The modified request is forwarded by WebSEAL to WebSphere.
6. The plug-in, *TrustAssociationInterceptor* (TAI), establishes that WebSphere is going to trust the WebSEAL server. It uses the *validateEstablishedTrust* method to do this. Optionally, it will bind the WebSEAL user name and password to the LDAP server for authentication.
7. The plug-in extracts the end user's user name from the iv-user header field and passes it to WebSphere to handle authorization.

The steps needed to enable Web Trust Association are as follows:

1. Configure WebSphere to enable Trust Association.
2. Set up the Trust Association Interceptors that will receive the request from the RPSS.
3. Configure an SSL WebSEAL junction.
 - a. Configure Web Trust Association for WebSphere. From the WebSphere Administration Console, open the Security Center. Go to the Authentication tab and select **Enable Web trust association**.

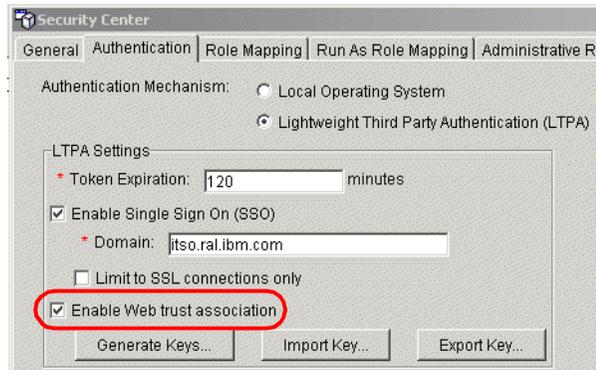


Figure 13-25 Configuring WebSphere for trust association

Click **OK** to save your changes.

- b. There are two associated properties files: *trustedservers.properties* and *webseal36.properties*. These files are located in the `<WAS_HOME>\properties` directory.

The file *trustedservers.properties* specifies the type of RPSS that is being used (in this case, *webseal36* is the type). It also specifies the class name in which the interceptor is implemented and the name of the second properties file.

The example of this file (see Example 13-2) contains the default entries as shipped with WebSphere. You should not need to modify it for this example.

Example 13-2 *trustedservers.properties* file for WebSEAL

```
# Trust Association Properties
# IBM WebSphere Application Server, Version 4.0, 2001

#Use this property to specify the types of reverse proxy
#servers that will be loaded at runtime
com.ibm.websphere.security.trustassociation.types=webseal36
```

```

#For each type of reverse proxy servers specified in
#com.ibm.websphere.security.trustassociation.types,
#specify the class file that implements the associated
#interceptor for it.
com.ibm.websphere.security.trustassociation.webseal36.interceptor=com.ibm.ws.se
curity.web.WebSealTrustAssociationInterceptor

#Optionally, specify a properties file for any of the
#reverse proxy servers type specified above. The properties file
#must end with ".properties". e.g. webseal36.properties. However,
#do not include this extension as shown below. Moreover, you can only
#do this if the interceptor class extends
WebSphereBaseTrustAssociationInterceptor
#and both init() and cleanup() methods were implemented. The init()
#method should read and parse the contents of the properties file.
com.ibm.websphere.security.trustassociation.webseal36.config=webseal36

```

The file *webseal36.properties* specifies the following:

- Special header fields that will be sent by WebSEAL with the request to WebSphere.
- The location of the WebSEAL server.
- The port where WebSEAL will receive the user requests.
- The user name that WebSEAL must use to establish trust with WebSphere.

The version of this file that is shipped with WebSphere V4 needs to be modified for your environment. Uncomment all the directives and change the values as follows:

Table 13-6 Properties for *webseal36.properties* file

Property	Value
com.ibm.websphere.security.webseal36.id	iv-user
com.ibm.websphere.security.webseal36.hostnames	<WebSEALHostname>
com.ibm.websphere.security.webseal36.ports	443
com.ibm.websphere.security.webseal36.loginId	WebSeal

Example 13-3 shows the version of this file that we used in this scenario.

Example 13-3 *webseal36.properties* file

```

# WebSeal 3.6 Trust Association Interceptor Configuration file
# IBM WebSphere Application WebSphere Version 4.0, 2001

#Uncomment and use this property to specify the header name(s)
#you expect to exist in the HTTP Request

```

```
com.ibm.websphere.security.webseal36.id=iv-user
```

```
#Uncomment and use this property to specify where you expect the WebSeal 3.6  
#server(s) to be.
```

```
com.ibm.websphere.security.webseal36.hostnames=m23vnx64.itso.ral.ibm.com,  
m23vnx64
```

```
#Uncomment and use this property to specify the port(s) from which the WebSeal  
#3.6 server(s) receive user requests
```

```
com.ibm.websphere.security.webseal36.ports=443
```

```
#Uncomment and use this property to specify the id that the webseal server must  
#use to validate trust with the interceptor
```

```
com.ibm.websphere.security.webseal36.loginId=WebSeal
```

- c. For Web Trust Association, the communication between WebSEAL and the Web server must use SSL so that WebSEAL can authenticate the Web server. This means that both WebSEAL and IHS must be configured for SSL. IHS must be configured as shown in “Configuring the Web Server to support HTTPS” on page 239. The keyring that WebSEAL uses must contain the CA’s root certificate for the server certificate that IHS is using.

This SSL link can use either self-signed certificates or CA-signed certificates. Either way, the signer of the certificate has to be included in WebSEAL’s keyring file. We used a self-signed certificate in this example.

Using the instructions in “Configuring the Web Server to support HTTPS” on page 239, create a keyring file (.kdb) for IHS to use for SSL.

Important: It is important, when importing the IHS root CA certificate into the Policy Director keyring, that you use the ikeyman tool that comes with the GSKit. You installed the GSKit during the Policy Director installation; the default location is c:\program files\ibm\gsk4\bin\gsk4ikm.exe. Due to differences in the levels of the GSKit used by IHS and PD, the SSL connection between WebSEAL and IHS will be impossible to establish if you use the IHS version.

- i. Start the ikeyman tool that comes with the GSKit.
- ii. Open the keyring file (**Key Database File -> Open**), then choose the file and click **Open**.
- iii. Export the signer certificate.

As we are using a self-signed certificate, make sure that you are in the Personal Certificates view. Select your certificate and click **Extract Certificate....**

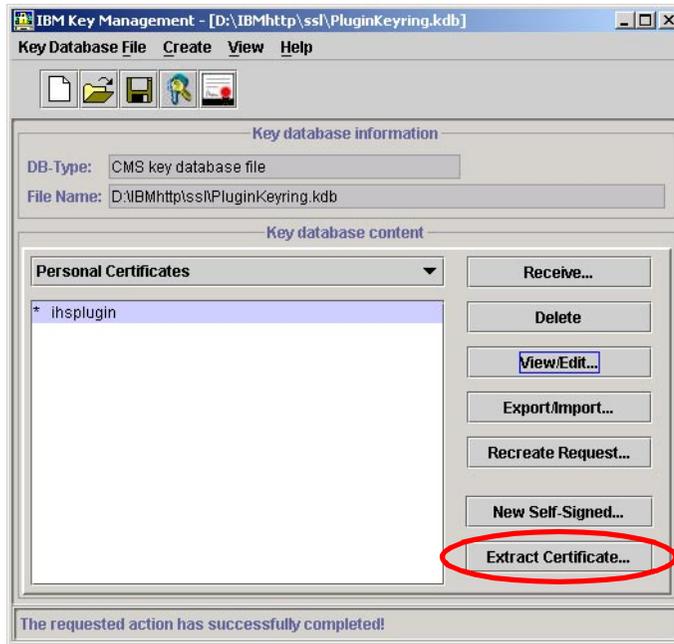


Figure 13-26 Extracting a self-signed certificate

- iv. You will be presented with the Extract Certificate to a File pop-up window. Fill out the fields, then click **OK**.

Data type: Base64-encoded ASCII data

Certificate file name: cert.arm

Location: C:\IBM HTTP Server\

Note: If you are using a CA-signed certificate, then switch to Signer Certificates; select the CA root certificate and click **Extract**. You will be presented with the Extract Certificate to a File pop-up window where you can save the certificate.

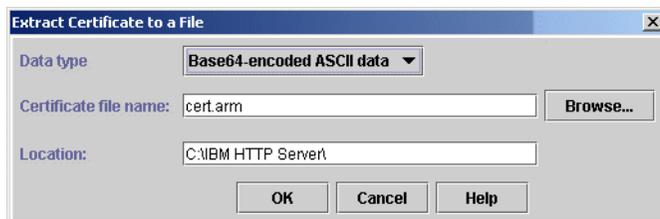


Figure 13-27 Exporting CA root certificate

- v. Close this keyring by clicking **Key Database File -> Close**.
- vi. Open the WebSEAL keyring file
`<PD_HOME>\PDWeb\www\certs\pdsrv.kdb`. The password for this file is *pdsrv*.
- vii. Switch to Signer Certificates and click **Add**.
- viii. Enter the name and location of the CA's root certificate you saved, then click **OK**. When prompted, enter a label for the certificate, for example, IHS Root Certificate.
- ix. Close this keyring file.

Note: Make sure, if you created the keyring file elsewhere, that you copy the stash file (.sth) to the same location as the .kdb file.

- d. To configure the Web server to accept encrypted traffic (HTTPS), we must modify the configuration file for IHS. Add the following lines to the `httpd.conf` file:

```
LoadModule ibm_ssl_module "<IHS directory>/modules/IBMModuleSSL128.dll"
AddModule mod_ibm_ssl.c
Listen 443
Keyfile <IHS keyring file>
SSLV2Timeout 100
SSLV3Timeout 1000
NameVirtualHost <fullHostname:443>
<VirtualHost <fullHostname:443>>
SSLEnable
SSLClientAuth none
SSLServerCert <yourCertificateName>
ServerName <yourServerName>
DocumentRoot "c:/ibm http server/htdocs"
</VirtualHost>
```

Below are the entries we used in our httpd.conf file.

Example 13-4 Our httpd.conf file entries for SSL

```
LoadModule ibm_ssl_module "c:/ibm http server/modules/IBMModuleSSL128.dll"
AddModule mod_ibm_ssl.c
Listen 443
Keyfile "c:/ibm http server/PluginKeyring.kdb"
SSLV2Timeout 100
SSLV3Timeout 1000
NameVirtualHost m23caaah.itso.ral.ibm.com:443
<VirtualHost m23caaah.itso.ral.ibm.com:443>
SSLEnable
SSLClientAuth none
SSLServerCert ihsplugin
ServerName m23caaah.itso.ral.ibm.com
DocumentRoot "c:/IBM http server/htdocs"
</VirtualHost>
```

Note: Because we have WebSEAL and the Web server on two separate machines, we can use port 443 for the secure traffic on both of them. If everything were installed on a single box, you would use different ports for the SSL traffic to avoid conflicts.

Verify the SSL configuration by pointing your browser to the Web server machine (<https://<fullHostname>>). You may get a security warning, then you should see the IHS home page.

- e. Now we need to create the WebSEAL SSL junction to the Web server. This junction needs to support the special header in which the client's user name will be transmitted, *iv-user*. We also need to define the special WebSEAL user name that will be used to authenticate the server with WebSphere.

- i. First, create the new WebSEAL user and make it active. We created a user called WebSeal.

```
pdadmin> user create WebSeal cn=WebSEAL,c=US,o=webbank WebSeal
WebSeal passwd
```

```
pdadmin> user modify WebSeal account-valid yes
```

- ii. Create the SSL junction. The syntax is:

```
pdadmin> server task <WebSEALServer> create -t ssl -h
<IHSFullHostname> -i -j -B -U <username> <password> -c iv-user
</junctionName>
```

In our example, this translated to:

```
pdadmin> server task webseald-m23vnx64 create -t ssl -h  
m23caaah.itso.ral.ibm.com -i -j -B -U WebSeal passw0rd -c  
iv-user /wastai
```

- f. Restart the WebSphere admin server and make sure the Webbank application is running. Now, point your browser to `https://<WebSEALHostname>/wastai/webbank`. If you enter a user name that does not have authority to access the Webbank application, you will get a You are not authorized to view this page message, as with the LTPA authentication. With a user name from the Employee or Manager role, if you have kept the SSO flag on, you will see an LTPA cookie returned.

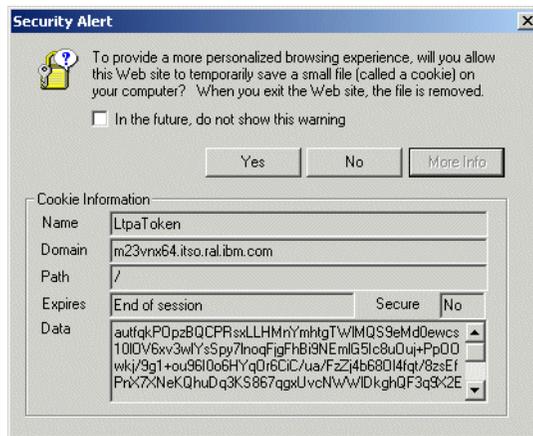


Figure 13-28 LTPA token as a cookie with SSO and Web Trust Association

The SSO flag is on in WebSphere because our Webbank application used Form-based authentication, which requires SSO. Now that Policy Director is providing the authentication, this flag is no longer required.

Accept this cookie and you will be presented with a second cookie for the junction.

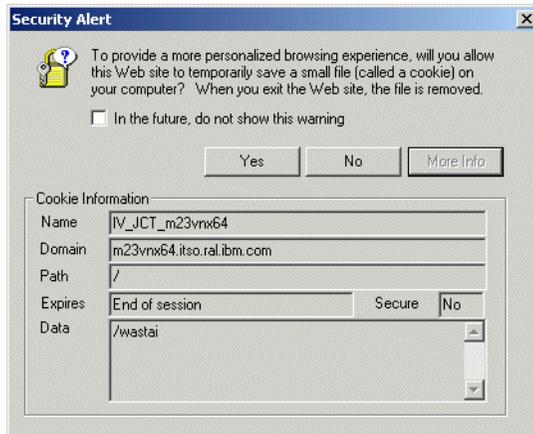


Figure 13-29 Web Trust Association junction cookie

Accept this cookie and you will be presented with the Webbank home page.

If you turn on tracing for security in the Administration Console, then start a new browser and go to the Webbank application, you can see what is happening by looking in the tracefile.

Note: These are not continuous trace sections; rather, we have extracted the key lines from the trace to show you what is happening.

1. First, the WebSEAL identity is authenticated:

Example 13-5 WebSEAL user name being authenticated by WebSphere

```
[01.11.05 21:20:54:667 EST] 56551f4f LdapRegistryI D Authenticating
                             WebSeal
[01.11.05 21:20:54:667 EST] 56551f4f LdapRegistryI D Searching for users
[01.11.05 21:20:54:677 EST] 56551f4f LdapRegistryI > getUsers
                             WebSeal
[01.11.05 21:20:54:677 EST] 56551f4f LdapRegistryI D filter =
                             (&(uid=WebSeal)(objectclass=ePerson))
[01.11.05 21:20:54:727 EST] 56551f4f LdapRegistryI D Found user
                             cn=WebSEAL,c=US, o=webbank
```

2. Then WebSphere maps the end user identity and this WebSEAL identity. We used the user name *joanna* to log in to the Webbank application.

Example 13-6 WebSphere maps the end user to WebSEAL identity

```
[01.11.05 21:20:55:118 EST] 56551f4f LTPAServerObj > mapCredential
[01.11.05 21:20:55:118 EST] 56551f4f LTPAServerObj D TrustAssociation id in the
Credential
[01.11.05 21:20:55:118 EST] 56551f4f LTPAServerObj >
mapTrustAssociationUserCredential
[01.11.05 21:20:55:138 EST] 56551f4f SecurityColla D
MethodName=findAllByPattern(java.lang.String):0
AllByPattern
[01.11.05 21:20:55:148 EST] 56551f4f RegistryEntry D pattern
joanna
```

3. Now, WebSphere checks that this user is authorized to access the requested resource.

Example 13-7 WebSphere authorizes the end user

```
[01.11.05 21:20:55:358 EST] 56551f4f SecurityColla > performAuthorization
[01.11.05 21:20:55:428 EST] 56551f4f WSRegistryImp > getPrivilegeAttributeId
[01.11.05 21:20:55:428 EST] 56551f4f WSRegistryImp D user
cn=Joanna Hodgson,c=US, o=webbank
[01.11.05 21:20:55:569 EST] 56551f4f LTPAServerObj D accessId :
user:m23vnx64.itso.ra1.ibm.com:389/cn=Joanna Hodgson,c=US, o=webbank
[01.11.05 21:20:55:619 EST] 56551f4f SecurityColla D AccessAllowed true
```



Single Sign-On

This chapter discusses Single Sign-On with the IBM WebSphere Application Server V4 Advanced Edition.

There are different scenarios in which Single Sign-On is exercised between application servers:

- ▶ WebSphere Application Server-WebSphere Application Server
- ▶ WebSphere Application Server-Lotus Domino server, using Secureway LDAP
- ▶ WebSphere Application Server-Lotus Domino server, using Domino LDAP

14.1 Single Sign-On

Single Sign-On (SSO) is the process which permits Web users to move between different applications located on the same or different physical machines, without being prompted for a user name and password (or certificate) every time.

The Lightweight Third Party Authentication (LTPA) mechanism is used for enabling SSO between servers. This mechanism utilizes an *LTPAToken* which contains the user authentication information, the network domain in which the SSO should be valid and the expiration time after which the user might be required to re-authenticate. The LTPAToken is encrypted using the LTPA keys that must be shared between all the SSO participating servers.

The token is issued to the Web user in a cookie called a *transient* cookie; this means that the cookie resides in the browser memory, is not stored on the user's computer system and expires when the user closes the browser. This cookie is easily recognized by its name: *LtpaToken*.

It is possible to enable SSO between WebSphere servers, between Domino servers and between WebSphere and Domino servers.

The requirements for enabling SSO are:

- ▶ The use of the same registry (LDAP Server) for authentication.
- ▶ All SSO participating servers must be in the same DNS domain.
- ▶ The URLs must include the DNS domain (no IP addresses or host names).
- ▶ The browsers must be configured to accept cookies.
- ▶ Servers' time and time zone must be correct (SSO token expiration time is absolute).
- ▶ All servers must share the LTPA keys.

The purpose of this section is to explain how to configure the SSO between WebSphere and Domino. To demonstrate SSO, we used the Webbank application and created a new Domino database to allow users to send comments to the bank site. For more information about the sample application, refer to "Domino Webbank sample" on page 98. These two applications are presented in the following scenarios:

1. WebSphere-Domino using IBM SecureWay as the common LDAP registry.
2. WebSphere-Domino using IBM SecureWay as the common LDAP registry with SSL.
3. WebSphere-Domino using Domino as the common LDAP registry.
4. WebSphere-Domino using Domino as the common LDAP registry with SSL.

14.2 WebSphere-Domino using SecureWay Directory

We used the following product versions in the example documented below:

- ▶ Windows 2000 with Service Pack 2.
- ▶ 1 GB RAM and 16 GB of hard disk space.
- ▶ IBM SecureWay Directory Server V3.2.1 for Windows.
- ▶ IBM DB2 Universal Database V7.2.1, Enterprise Edition for Windows.
- ▶ IBM HTTP Server 1.3.19 for Windows as the Web server installed in the same machine as the WebSphere server.
- ▶ Lotus Domino Server R 5.06a.
- ▶ WebSphere Application Server Advanced Edition 4.01.
- ▶ Microsoft IE 5.5 for the Web user.

The example is shown in Figure 14-1:

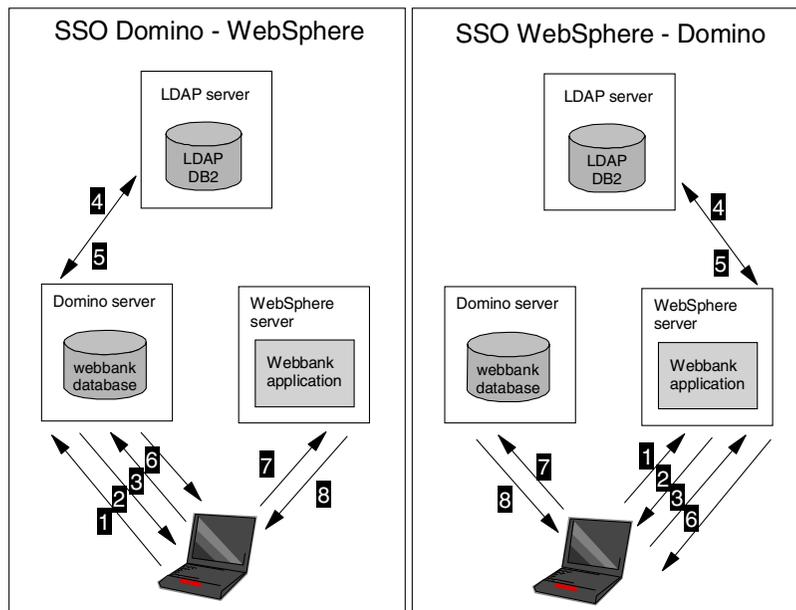


Figure 14-1 Sample SSO Domino - WebSphere using IBM SecureWay Directory

1. A Web user submits a request to a Web server (Domino or the HTTP Server) for a protected resource. In the case of Domino, the user wants to create a new comment in the *webbank.nsf* database. In the case of WebSphere, the user wants to make a new bank transfer.

2. The Web server prompts the user for the authentication information.
3. The user responds to the challenge by supplying the information (user name and password or certificate).
4. The Web Server contacts the LTPA server (Domino or WebSphere) which connects with the IBM SecureWay Directory to verify the authentication information.
5. If the information supplied for the user is correct, the IBM SecureWay Directory responds to the LTPA server with the validated information.
6. The LTPA server uses the returned values to check whether the user has access to the requested resource and issues an LTPA token for the user. The Web server sends the token to the user as an HTTP cookie, which is stored in the user's browser, and serves the requested resource (opening the Webbank database in the case of Domino or *Webbank.html* in the case of WebSphere).
7. Once the user is authenticated and has the cookie available, he or she can request another protected resource to Domino or WebSphere.
8. Domino or WebSphere validates the token provided for the user and tells the Web server to send the requested resource to the browser (as long as the user has access to that resource) without prompting again for user information.

The necessary steps to set up Single Sign-On between WebSphere and Domino involve:

- ▶ Installing and configuring software products and application examples.
- ▶ Enabling Single Sign-On for WebSphere Application Server.
- ▶ Enabling Single Sign-On for the Domino Server.

Installing and configuring software products and applications

For more details on how to install and configure the software products, refer to the installation manuals; in the case of IBM Secureway Directory Server, refer to Chapter 16, "IBM WebSphere Application Server and LDAP" on page 459.

Make sure, before you begin enabling SSO, that the Webbank application has been installed in WebSphere and that the Webbank Comment Application database has been created in Domino. For more details, refer to Section 5.6, "Domino Webbank sample" on page 98.

Directory structure

In our example, we set the suffix o=Webbank, where o represents an organization. Then we created a new Organizational Unit ou=ITSO and added the Users and Groups beneath this organizational unit. The directory structure is shown in Figure 14-2.



Figure 14-2 IBM SecureWay Directory structure

14.2.1 Enabling Single Sign-On for WebSphere

To set up SSO in WebSphere for our example, it is necessary to execute the following tasks:

- ▶ Configure the Global Security Settings in WebSphere for SSO.
- ▶ Apply security to the Webbank application example.

Configuring Global Security Settings for SSO

To configure Global Security Settings for SSO in WebSphere, perform these main configuration processes:

1. Start WebSphere Administrator's Console (the WebSphere administration Server 4.0 must be running first).
2. Select **Console -> Security Center...** This will display the Global Security Settings for WebSphere. Select **Enable security** in the General tab.
3. Click the **Authentication** tab and choose **Lightweight Third Party Authentication (LTPA)** as the Authentication mechanism type.
4. Specify the following LTPA settings:
 - Select how many minutes can pass before a client using an LTPA token must authenticate again, in the **Token Expiration** field. By default, this is 120 minutes.

- Select **Enabled Single Sign-On (SSO)**. The Domain field will be currently enabled.
- Enter a DNS domain name in the **Domain** field. In our example, we set this domain to `its0.ral.ibm.com`. This domain name is used when the HTTP cookie is created for SSO and determines the scope to which SSO applies.

Important: Remember that all SSO participating servers must be in the same DNS domain.

5. Select the **LDAP** radio button and introduce the LDAP server settings as explained in Chapter 16, “IBM WebSphere Application Server and LDAP” on page 459.

All these settings are illustrated in Figure 14-3.

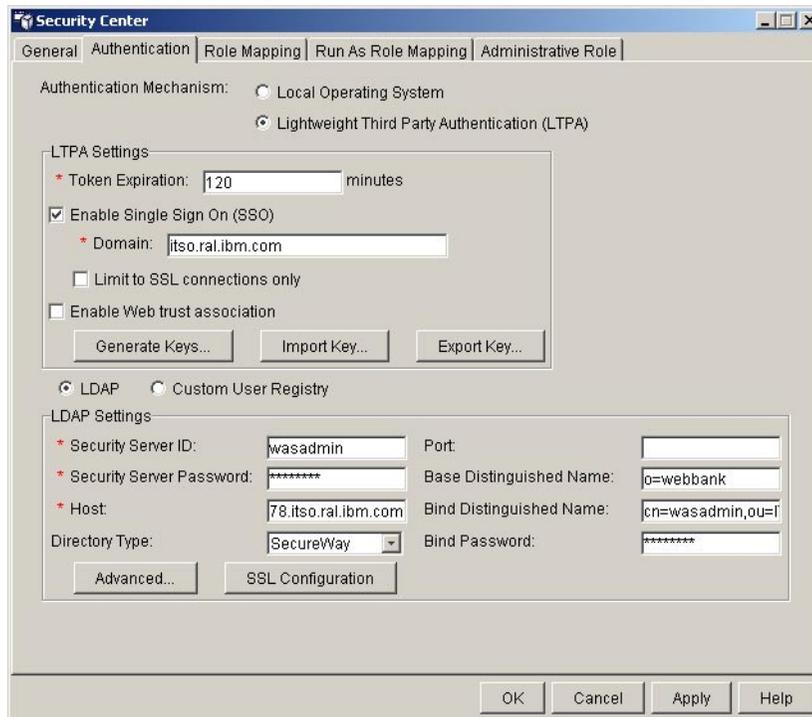


Figure 14-3 Specifying LTPA authentication and SSO in WebSphere

Note: Selecting the **Limit To SSL Connections only** field allows you to use a connection with SSL for SSO. We discuss this topic later in this chapter.

6. Click the **Generate Keys...** button to create the LTPA keys for encrypting the LTPA token. You will be prompted for an LTPA password to protect the set of encryption keys; type in your password, for example websphere. The LTPA keys must be shared between all the servers using SSO.
7. Click **OK**. A new message in the Administrator's Console will appear:
Command 'Generate LTPA Keys" running
When the process is completed, the Console will show the following message:
Command 'Generate LTPA keys" completed successfully
8. Once the LTPA keys are generated, click the **Export Key...** button to export the LTPA keys to a file. We will use this file to import the keys in Domino. Save your file to wassecurew.keys.
9. Choose the directory where you want to save the file and give it a name. Click **Save**. An example of this file is shown in Example 14-1.

Example 14-1 wassecurew.keys file

```
#
#Mon Nov 05 17:10:54 EST 2001
com.ibm.Websphere.CreationDate=Mon Nov 05 17\:\:10\:\:54 EST 2001
com.ibm.Websphere.ltpa.PublicKey=ALTD13oj+ShmKL3FeYxucbociBn6avPaqeGwtGwekKmcOif3tK6gaTIOsvrIkAkkm2X04915SCJePfhI8DpX0aHDPJx4sz9Y4FifT2zq8RIPESg/dDdz+DwZUAamX4tbtqp1FaCD4LxkcEQSkm9L9SC6jSHaQqaM/1YeAtdoLNhtAQAB
com.ibm.Websphere.ltpa.version=1.0
com.ibm.Websphere.ltpa.Realm=M23VNX78.itso.ra1.ibm.com
com.ibm.Websphere.ltpa.PrivateKey=0No+8v7dhnJ78WtzAhyZIV6imchM9E20Rj7AJM8L5MCTZIWZAHAzU6GuNNn0e51TdR60YXF9X/Kiph2Psij4q32C67Kbo9279WEpJp7+0b7RYXz7Ah/Hq71Y6fqyznY/aHTS98BzAFyp/vLbUpo/15gapI1HYnciWJ9VM4a9BCXEGkV+L7BNw3UqpAzsNb3ncolM8UoHADz+7A/72Xr8fjUtDpXJZKwvhOM+kgVMhqokf1TnxJx84AtqRC7ayDMFxLx+ee9c0DHHljVxcTUc57YIrw8hKazsh2xn7+3f/V6QPndotemJzJdd56EIupQkEEkRcZRwl9JEh5awrM9aTcCOZpTo68o8synbkV9Vw/g\=
com.ibm.Websphere.CreationHost=M23CABYG
com.ibm.Websphere.ltpa.3DESKey=TEDMXeMc2WF9IGTFvLK5I8UD4eyLaB9Ep7ji+BJPSDM\=
```

As we can see in the example, three types of LTPA keys have been generated:

- The private key: used for the LTPA server to sign the LTPA Token.
- The public key: used to verify the digital signature.
- A shared key: used to encrypt/decrypt those tokens.

Important: For security reasons, it is recommended that you generate a new set of keys periodically.

Note: The generation of the LTPA keys must be performed when the LDAP server settings are configured; this guarantees that the LDAP host name is present in the exported file, as shown in bold in the previous example. Domino needs this information during the Web SSO Configuration Document creation process.

10. Click **Apply** and then **OK**. Make sure that the IBM SecureWay Directory is running, because the Security server ID and Security Server Password will be verified against the LDAP server.
11. When the process is completed, a warning message will be displayed, stating: Changes will not take effect until the admin server is restarted. Click **OK**.
12. Restart the administration server by selecting the node in the node folder located in the tree view on the left side of the Console, right-clicking it and selecting **Restart** in the resulting context menu.

When the administration server is open for e-business, check that the security configuration is consistent and that the following lines are added in the `sas.server.props` file located in the `<WAS-HOME>\properties` directory:

Example 14-2 sas.server.props file Security/Authentication section

```
#Mon Nov 05 12:27:02 EST 2001
com.ibm.CORBA.loginPassword={xor}Lz4sLCgwLTs\=
com.ibm.CORBA.principalName=M23VNX78.itso.ral.ibm.com/wasadmin
com.ibm.CORBA.securityEnabled=true
com.ibm.CORBA.loginUserId=wasadmin
com.ibm.CORBA.authenticationTarget=LTPA
```

14.2.2 Enabling Single Sign-On for Domino

The SSO setup in Domino, for our example, involves the following tasks:

- ▶ Configuring Domino to use the IBM SecureWay Directory server as its user registry.
- ▶ Enabling SSO for Domino.
- ▶ Implementing the security to the `webbank.nsf` database.

Configuring Domino to use IBM SecureWay Directory

To authenticate Web users using the credentials included in the IBM SecureWay Directory, perform the following steps:

1. Create the Directory Assistance Database.

Domino uses this database to perform searches in other LDAP-compliant directories (such as secondary Domino directories or other LDAP Directory Servers like IBM SecureWay Directory).

To create a new Directory Assistance database, start the Domino R5 Administration client with a Notes administrator ID, then from the Domino Administrator menu:

- a. Choose **File -> Database -> New**. The new database Dialog Box is displayed.
- b. Select the server where you want to create the new database.
- c. Enter a title for the database, for example, Directory Assistance.
- d. Enter a file name for the database, for example, da.NSF.
- e. Click the **Template Server...** button and select the Domino server that stores the Directory Assistance template (DA50.NTF); highlight it.
- f. Make sure that **Inherit future design changes** is selected, then click **OK**.

This settings are illustrated in Figure 14-4.

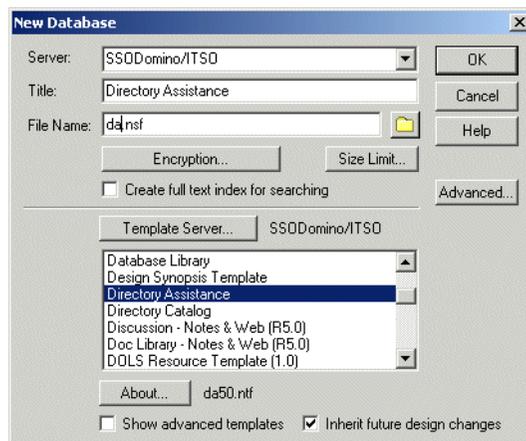


Figure 14-4 Creating the Directory Assistance database

Note: Create a replica of the Directory Assistance database if more Domino servers are going to use it.

- Identify the database on the server by opening the Domino server from the left server bookmark pane; click the **Configuration** tab, and select **Server -> Current Server document**. In the **Basics** tab, edit the document and include the name of the database created (da.nsf) in the Directory Assistance database name field as shown in Figure 14-5.

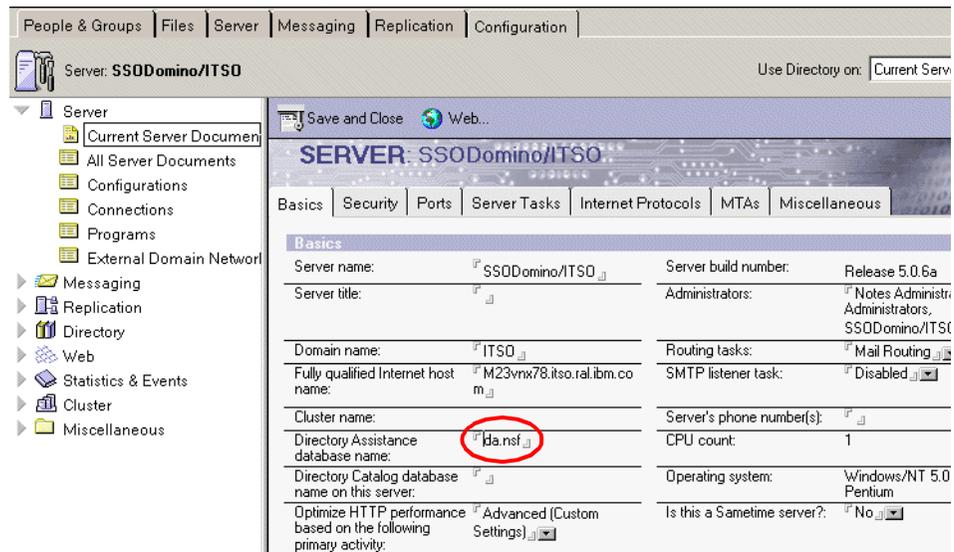


Figure 14-5 Identify Directory assistance Database on the server

- Save the document.
- In the same place, select **Directory -> Directory Assistance database** and click the **Add Directory Assistance** button to create a new Directory Assistance document. In the **Basics** tab, complete these fields:
 - Select **LDAP Directory** in the Domain Type field.
 - Enter the name of the Domain this record describes. The Domain name must be unique. In our example, we set it to SecureWay.
 - Select the name of the company associated with this directory (for example: IBM).
 - In the Search Order field, enter the number representing the order in which this directory is searched relative to other directories in the Directory Assistance Database. In our case, this is the only Directory Assistance document, so we do not need to add any number.
 - Choose **Yes** in the Group Expansion field to allow Directory Assistance to verify Web user membership when a group is included in a database ACL that the Web user is attempting to access.

- Choose **Yes** (the default) in the Nested Group Expansion field to allow servers to look up names in groups nested within LDAP directory groups, in database ACLs.
- Choose **Yes** in the Enabled field to enable directory assistance for this directory.

These settings are shown in Figure 14-6.

DIRECTORY ASSISTANCE	
Basics	Rules
LDAP	
Basics	
Domain type:	LDAP
Domain name:	SecureWay
Company name:	IBM
Search order:	
Group expansion:	Yes
Nested group expansion:	Yes
Enabled:	Yes

Figure 14-6 Basics tab in the Directory Assistance database

5. Click the **Rules** tab to specify one or more naming rules that correspond to the hierarchical names of entries in the directory. Directory assistance uses naming rules to determine the order in which to search directories when users provide hierarchical names.
 - a. For our example, we set the rule `*/**/*/*/*` to search for all names in the directory.

Note: For more information about Naming rules in Directory Assistance, refer to the *Domino R5 Administration Help and Administrator's Guide*.

- b. Choose **Enable** to implement this specific rule.
- c. In the **Trusted for Credentials** field, choose **Yes** to allow Domino to authenticate only Web clients with names that match the rule.

These settings are shown in Figure 14-7.

DIRECTORY ASSISTANCE								
Basics		Rules		LDAP				
- Use the first rule to configure the Base for this LDAP server (4.6x Servers only)								
Rules								
	OrgUnit4	OrgUnit3	OrgUnit2	OrgUnit1	Organization	Country	Enabled	Trusted for Credentials
Rule1	*/	*/	*/	*/	*/	*	Yes	Yes
Rule2	/	/	/	/	/		No	No
Rule3	/	/	/	/	/		No	No
Rule4	/	/	/	/	/		No	No
Rule5	/	/	/	/	/		No	No

Figure 14-7 Rules tab in Directory assistance

6. Click the **LDAP** tab and include the LDAP Configuration settings:
 - a. In the Host Name field, include the DNS Host name for the IBM SecureWay directory, for example M23VNX78.itso.ra1.ibm.com.
 - b. Enter a distinguished name in the Username field and a password in the Password field within the Optional Authentication Credential field. In our example, we created a new user in SecureWay with the following distinguished name (DN): cn=Domino Admin,ou=ITS0,o=Webbank

Note: The name and password must correspond to a valid name and password in the directory of the LDAP directory server. If you did not enter a name and password, the Domino server attempts to connect to the LDAP directory server anonymously.

We also recommend using a Notes secret encryption key to encrypt the Directory Assistance document so that only administrators with the encryption key can see the contents of the Username and Password fields; for more details, refer to the *Domino R5 Administration Help and Administrator's Guide*.

- c. Introduce the starting point for LDAP searches in the Base DN field, for example o=Webbank. This field is required for the SecureWay Directory.
- d. Select **Notes clients/Web Authentication** in the Perform LDAP search for field.
- e. Choose **None** in the Channel Encryption field to allow Domino Server to connect to the LDAP server without SSL.
- f. The port number, by default, is 389 if none is specified.
- g. In the Timeout field, enter the maximum number of seconds before a search is terminated. The default is 60 seconds.

- h. In the Maximum number of entries returned field, enter the maximum number of entries a single search can return; the default value is 100.

All these settings are illustrated in Figure 14-8.

LDAP Configuration	
Hostname:	M23VNX78.its.o.ral.ibm.com
Optional Authentication Credential:	
Username:	cn=Domino Admin,ou=ITSO,o=webbank
Password:	password
Base DN for search:	o=webbank
Perform LDAP search for:	<input checked="" type="checkbox"/> Notes Clients/Web Authentication <input type="checkbox"/> LDAP clients
Channel encryption:	None
Port:	389
Timeout:	60 seconds
Maximum number of entries returned:	100

Figure 14-8 LDAP tab in directory assistance

7. Save the document.
8. Make sure that the IBM SecureWay Directory is running by checking the list of Windows services on the SecureWay machine and using the TCP/IP **ping** utility to test the connection to the SecureWay Directory machine from the Domino Server machine.
9. Restart the Domino server by entering the **restart server** command in the Domino Console.

Enabling Single Sign-On for Domino

Setting up Single Sign-On for the Domino Server involves two main steps:

- ▶ Creating a Web SSO Configuration Document.
- ▶ Enabling Single Sign-On.

Follow these steps to enable SSO for Domino:

1. Create a new Web SSO Configuration Document in the Domino Directory database.

This action is only required *once* (it is only possible to create a Web SSO Configuration document *once* in your domain) and should be replicated to all servers participating in the Single Sign-On domain. This document is encrypted for all the participating servers and contains the LTPA keys used to authenticate user credentials.

- a. Open the Domino Directory and select **Server->Servers** to display the view. Click the **Web** button and select **Create Web SSO Document** in the resulting context menu.
- b. A new Document will be displayed with the *LtpaToken* Token Name field (). This name cannot be modified.
- c. Include the DNS domain in the Token Domain Field. This value must coincide with the value specified in the Domain field in WebSphere. This domain name is used when the HTTP cookie is created for Single Sign-On, and determines the scope to which Single Sign-On applies. For our example, we set this domain to `itso.ra1.ibm.com`.
- d. Choose the Domino servers that are going to participate in the SSO scenario (group names are not allowed), for example: `SSODomino/ITS0`.
- e. Enter the maximum number of minutes that the issued token will be valid in the Expiration (minutes) field. The Default is 30 minutes. We set this to 120 minutes to match it with the Token expiration time in WebSphere.
- f. Click the **Key...** Action and select **Import WebSphere LTPA Keys** from the drop-down menu.
- g. Specify the path and the file name for the WebSphere LTPA keys file exported previously.
- h. Click **OK**. A new dialog box will appear, prompting the user for the LTPA password specified when the keys were generated (see Figure 14-9).



Figure 14-9 Entering the LTPA password

- i. Click **OK**. When the process completes, a confirmation message will be displayed.
- j. A new WebSphere Information section will appear in the document, as shown in Figure 14-10.



Figure 14-10 WebSphere Information in the Web SSO Configuration Document

The *LDAP realm* is read from the WebSphere Import File and specifies the LDAP host name included in the WebSphere LDAP settings. This name must also coincide with that in the Host Name field specified in the LDAP configuration settings in the Directory Assistance database.

When a port is specified in the WebSphere LDAP configuration settings, it will be included in the LTPA key exporting file using the following format: M23VNX78.itso.ral.ibm.com\ :390. But when the LTPA keys are imported in Domino in the LDAP Realm Name, the backslash disappears: M23VNX78.itso.ral.ibm.com:390.

Make sure you add a backslash (\) prior to the colon (:) and replace the value above with the following: M23VNX78.itso.ral.ibm.com\ :390.

The LTPA version denotes the version of the WebSphere LTPA implementation. It is read from the LTPA importing file.

- k. Click the **Save and Close** button. The document will be saved. To check if the document is present in the Domino Director, select **Server -> Web Configurations View** and expand the ***- All Servers -** section. The new document created should be displayed as Web SSO configuration for LtpaToken.
2. Enabling Single Sign-On in the Domino Server document involves the following steps:
 - a. Make sure that in the Document server, the TCP/IP port status in the Web tab (**Ports -> Internet Ports -> Web**) is enabled and the Anonymous Authentication option is set to No, as shown in Figure 14-11.

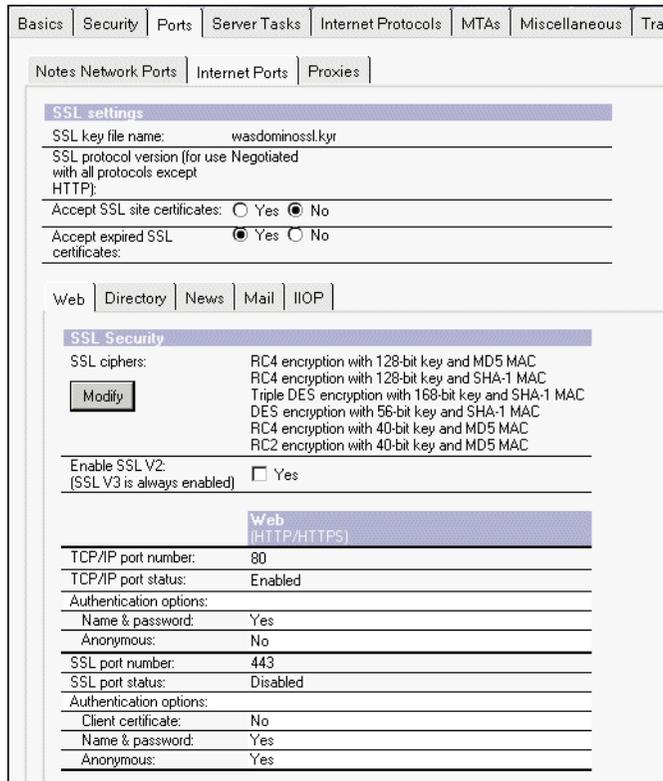


Figure 14-11 TCP/IP port status settings in the Server Document.

- b. Configure the Domino HTTP session support by selecting the **Domino Web Engine** tab (**Internet Protocols -> Domino Web Engine**) of the Domino server Document. Select the **Multi-Server** session. Selecting this option allows the Web user to log on once in the Domino Server and then gain access to another Domino Server or WebSphere server without logging on again.
- c. Open the Domino server document and switch to the **Security** tab. Go to the *Web server access* section and select **Fewer name variations with higher security** in the Web Server Authentication field.

Selecting this level of restriction makes Domino servers less vulnerable to security attacks by refining how Domino searches for names in the LDAP directory. This option requires users to enter the following user name formats in the user name and password dialog box, as shown in Table 14-1.

Table 14-1 User name formats

Using LDAP Directory for authentication
DN (Full Distinguished Name)
CN (Common Name)
UID or UID with UID= prefix

- d. Save the Domino Server Document.
- e. Restart the HTTP server task by entering the **tell http stop** and **load http** commands or the **Tell http restart** command in the Domino Console. A new message will appear in the console, as shown in Figure 14-12



Figure 14-12 Http: Successfully loaded Web SSO configuration

If a server enabled for SSO cannot find a Web SSO Configuration Document or if it is not included in the Server Names field of the document so the document cannot be encrypted, the following message should appear in the Domino Server console:

HTTP:Error Loading Web SSO configuration.Reverting to single-server session authentication.

14.2.3 Implementing the security to the Webbank.nsf database

To implement the security for the Webbank database, we modified the Database Access Control List (ACL), adding the LDAP Users or LDAP Groups to control the users' access to the database.

To add new user names or groups to the ACL, use the LDAP format for the name, but use forward slashes (/) as delimiters rather than commas (.). For example, if the name of a user in the LDAP directory is:

```
uid=vamor,ou=ITS0,o=Webbank
```

then you should enter in the database ACL:

```
uid=vamor/ou=ITS0/o=Webbank
```

To add the name of a non-hierarchical LDAP directory group in an ACL, do not include the name of an attribute as part of the entry, but only the value for the attribute. For example, if the name of the LDAP group is: cn=manager in the ACL, enter only: manager . However, if the name of the group is hierarchical in the ACL: cn=manager ,ou=ITS0,o=Webbank, then you should enter: cn=manager/ou=ITS0/o=Webbank .

Note: When the LDAP attributes correspond with the attributes used in Notes (for example: cn, ou, o, c) , the ACL will not display the attributes. For example: cn=manager/ou=ITS0/o=Webbank appears in the ACL as manager/ITS0/Webbank.

To add users and groups to the ACL database, make sure that you have manager access to the database and perform the following tasks:

1. Select the database icon from your bookmarks page.
2. Select **File -> Database -> Access Control**.

We set the following ACL for the Webbank Comment Application database:

Table 14-2 Webbank Comments Application database ACL with LDAP users

People, Servers and Groups	User Type	Access Level	Authorization
-Default-	Unspecified	No Access	None
LocalDomainServers	Server Group	Manager	Delete Documents
OtherDomainservers	ServerGroups	No access	None
Anonymous	Unspecified	No access	None
Administrators	People Group	Manager	Delete Documents
Manager/ITS0/Webbank	People Group	Editor	Delete Documents
Joanna Hodgson/ITS0/Webbank	Person	Editor	Delete Documents

All these settings are shown in Figure 14-13.

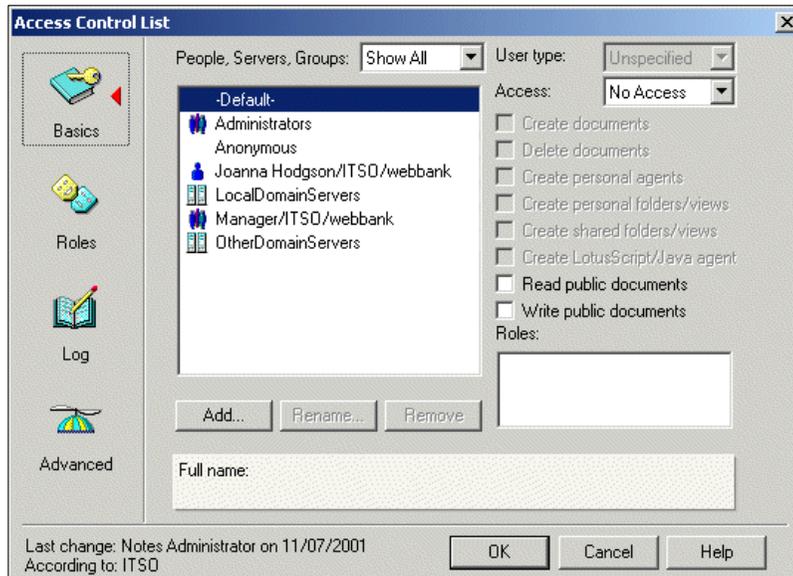


Figure 14-13 Webbank Comments Application database ACL

3. Click the **Advanced** button and set the *Maximum Internet name & password access* field to **Editor**, to allow a Web user to create documents in the database.

14.2.4 Testing Single Sign-On between Domino and WebSphere

We demonstrate the SSO between Domino and WebSphere in the following scenarios:

- Testing SSO between Domino and WebSphere
- Testing SSO between WebSphere and Domino

Testing SSO between Domino and WebSphere

The Web user wants to create a new Comment Document in the Webbank Comment application database, specifying the Webbank database URL in its browser. In our case, this is:

`http://m23x2640.itso.ra1.ibm.com/Webbank.nsf/Comments?OpenForm`

The Domino server will present a default server Login page, as shown in Figure 14-14.



Figure 14-14 Domino Server Session-based authentication prompt

The Web user has to type in the user name (uid attribute from LDAP) and password and click the **Login** button.

The Domino server will perform a search in the IBM SecureWay Directory Server looking for the user and verifying the password and the Web user group membership, as we can see in the Domino Console server:

Example 14-3 Domino searches messages in the Domino Console

```
> 11/07/2001 01:37:53.71 PM [ODE8:0047-0E60] WebAuth> LOOKUP user='vamor'
11/07/2001 01:37:53.71 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
name='vamor' in LDAP server='M23VNX78.ITSO.RAL.IBM.COM'
> 11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] WebAuth> BIND LDAP
host='M23VNX78.ITSO.RAL.IBM.COM:389' with name='cn=Domino
Admin,ou=ITSO,o=Webbank' and password='password', msgid='1'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'FullName'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'$$DomainType'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'$$DBIndex'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'AltFullName'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'AltFullNameLanguage'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'userPassword'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH Attr:'CN'
11/07/2001 01:37:53.73 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'ObjectClass'
```

```

11/07/2001 01:37:53.75 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Base=o=Webbank
11/07/2001 01:37:53.75 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH Scope=2
11/07/2001 01:37:53.75 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Filter=(|(cn=vamor)(uid=vamor)(sn=vamor)(givenname=vamor))
11/07/2001 01:37:53.75 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Timeout=60 secs
11/07/2001 01:37:53.75 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH returned
'1' match(es).
11/07/2001 01:37:53.75 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH returned
matched DN='cn=Victoria Amor/ou=ITS0/o=Webbank'
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] WebAuth> BIND LDAP
host='M23VNX78.ITS0.RAL.IBM.COM:389' w/ user='cn=Victoria
Amor,ou=ITS0,o=Webbank', password='password'
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] WebAuth> VERIFY password='password'
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] WebAuth> Get group info for new
user.
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] WebAuth> GroupCache: WildCard
Name='*'
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] WebAuth> GroupCache: Hierarchical
Name='*/ou=ITS0/o=Webbank'
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] WebAuth> GroupCache: Hierarchical
Name='*/o=Webbank'
11/07/2001 01:37:53.76 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
name='cn=Victoria Amor/ou=ITS0/o=Webbank' in LDAP
server='M23VNX78.ITS0.RAL.IBM.COM'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] WebAuth> BIND LDAP
host='M23VNX78.ITS0.RAL.IBM.COM:389' with name='cn=Domino
Admin,ou=ITS0,o=Webbank' and password='password', msgid='1'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'ListName'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'$$DBIndex'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'$$DomainType'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH Attr:'CN'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Attr:'ObjectClass'
11/07/2001 01:37:53.78 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Base=o=Webbank
11/07/2001 01:37:53.79 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH Scope=2
11/07/2001 01:37:53.79 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Filter=(|(&(objectclass=groupOfUniqueNames)(UniqueMember=cn=Victoria
Amor,ou=ITS0,o=Webbank))(&(objectclass=groupOfNames)(Member=cn=Victoria
Amor,ou=ITS0,o=Webbank)))
11/07/2001 01:37:53.79 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH
Timeout=60 secs

```

11/07/2001 01:37:53.79 PM [ODE8:0047-0E60] NameLookup|WebAuth> SEARCH returned '1' match(es).

11/07/2001 01:37:53.79 PM [ODE8:0047-0E60] NameLookup|WebAuth> **SEARCH returned matched DN='cn=Manager/ou=ITSO/o=Webbank'**

Note: To view the search messages in the Domino Console, include in the NOTES.ini file the following setting: WebAuth_Verbose_Trace=1. Use this setting only for troubleshooting problems with Web server user authentication and Web server group searches, because using it slows Web server performance.

The Domino server authenticates the user and finds that this user has editor access to the database, because it belongs to the Manager group included in the ACL. It will then create the LTPAToken and send it as an HTTP cookie, as shown in Figure 14-15.



Figure 14-15 LTPA token cookie

Note: To view the cookie, type the command **javascript:alert(document.cookie)** in the address bar (URL) of the Web browser.

The Domino server then opens the database and sends the user the following response:

New Comments - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History

Address <http://M23X2640.itso.ra1.ibm.com/webbank.nsf/Comments?OpenForm>

Webbank Comments

From : Victoria Amor

Subject: Comments

Provide **all your comments** below, and then Press **Submit**

My Comments are ...

Submit

Visit our Public site in <http://www.itsowebbank.com>

Visit our Secure site in <https://www.itsowebbank.com>

Figure 14-16 New comments document

At this point, the user can click the **Submit** button to save the document or return to the Webbank application home page by clicking the link **Visit our Public site in <http://www.itsowebbank.com>**; this link will connect the user to the WebSphere URL

(<http://m23cabvg.itso.ra1.ibm.com:90/Webbank/Webbank.html>, in our case) and will open the Webbank.html page without prompting you again for the user name and password, as shown in Figure 14-17.

Note: The WebSphere Application Server and Domino server are running on the same machine, and both of them are running the HTTP server. In order to avoid port conflicts, WebSphere was set to use port 90 for HTTP requests.

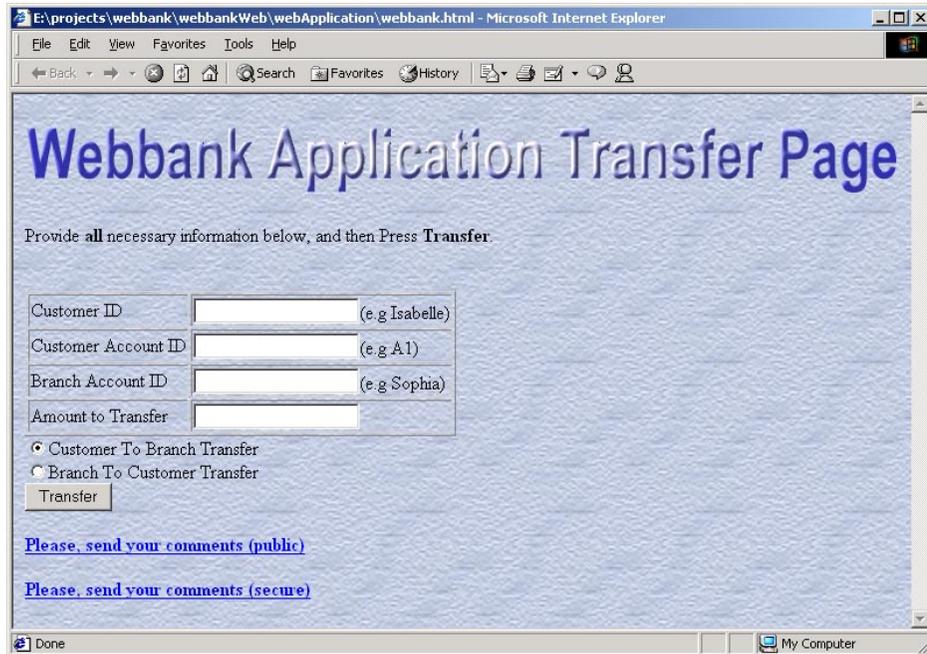


Figure 14-17 *Webbank.html page*

When the new Comments document is saved, it will be kept in the database as shown in Figure 14-18.

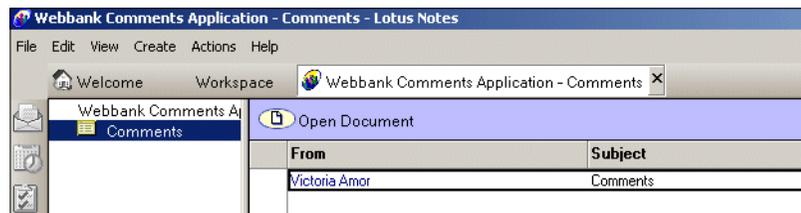


Figure 14-18 *New Comments document in the Webbank database*

Testing SSO between WebSphere and Domino

In this case, the user wants to make a new bank transfer specifying the Webbank application URL in its browser. In our case, this is:

`http://m23cabyg.itso.ra1.ibm.com:90/Webbank/Webbank.html`

The HTTP server prompts the user for the authentication informatio, as shown in Figure 14-19.

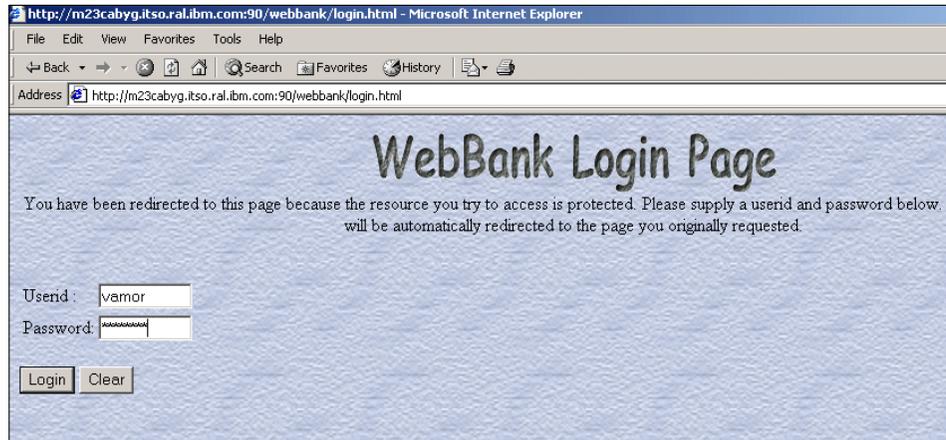


Figure 14-19 WebBank Login Page

The user responds to the challenge by supplying the information (user name and password) and clicking the **Login** button.

WebSphere will connect with the IBM SecureWay Directory Server to verify the authentication information. If the information supplied is correct, the directory server responds to WebSphere with the valid information.

WebSphere uses the returned values to check whether the user has access to the requested resource (Webbank.html) and issues an LTPA token for the user. The Web server sends the token to the user as an HTTP cookie as shown in Figure 14-15 on page 414, then opens the Webbank.html page as shown in Figure 14-17 on page 416.

At this point, the user can type in the data for the transfer and submit it, or send a new comment to the bank site by clicking the link **Please, send your comments (public)**. The link will connect to the Domino server URL (in our case <http://m23x2640.itso.ra1.ibm.com/Webbank.nsf/Comments?OpenForm>) and will open the Webbank Comments Application database as shown in Figure 14-16 on page 415, without prompting for the user name and password.

14.3 WebSphere-Domino using SecureWay LDAP with SSL

Here, we demonstrate the use of SSO in a secure environment using the IBM SecureWay Directory Server as the user registry. For that purpose, we have secured the communications between:

- ▶ The Web Browser and the Domino server.
- ▶ The Web Browser and the HTTP server.
- ▶ The Web Server (plug-in) and the WebSphere Application Server.
- ▶ Domino and the LDAP server.
- ▶ WebSphere and the LDAP server.

Before learning how to configure SSO in a secure environment, make sure that the following steps have been taken:

1. Certificates have been created for:
 - the LDAP server
 - the HTTPServer
 - Domino
 - the Web Server plug-in
 - WebSphere
2. The LDAP server has been configured to use SSL.
3. The IBM HTTP Server has been configured to use SSL.
4. The Web Server plug-in has been configured to use SSL.
5. WebSphere has been configured to use SSL.
6. Domino has been configured for SSL.

For more details of how to configure SSL and create the certificates, refer to Section 11.2, “Configuring the Web Server to support HTTPS” on page 239.

To enable SSO in a secure environment, do the following:

- ▶ Enable SSO in WebSphere to use SSL.
- ▶ Enable SSO in Domino to use SSL.

14.3.1 Enabling SSO to use SSL in WebSphere

Follow the instructions explained in “Enabling Single Sign-On for WebSphere” on page 397. Open the Security Center from the WebSphere Administrator’s Console, switch to the Authentication tab, then select **Limit To SSL Connections only**, which permits the use of a connection with SSL for Single Sign-On (SSO), as shown in Figure 14-20.

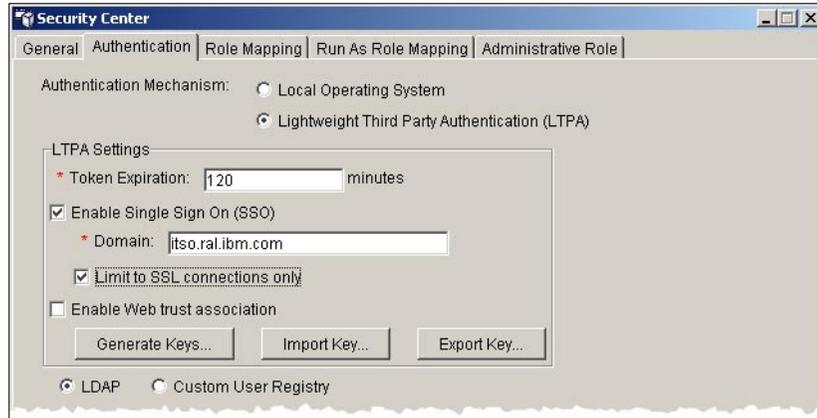


Figure 14-20 Selecting Limit to SSL Connections Only

Also remember to generate the LTPA keys after the configuration of the LDAP settings; this guarantees that the LDAP host name and port (636) are present in the exported file, as shown in Example 14-4.

Example 14-4 *wassecureSSL.keys*

```
com.ibm.WebSphere.CreationDate=Thu Nov 08 12\:\:18 EST 2001
com.ibm.WebSphere.ltpa.PublicKey=ANYPLyqpEjKejf6Vvgn6V2gbN9NmPEywfXTCw1c2aECXhx
/RtAP7BUt5DtV6Edg+ZA0tr50QnbyAxOYC2z9dIF+ItN0mSyAYu31ot1/zn1I44KSJGi1J7hoe0oK/q
HWP884MnIEIIsa44pqykb7DT0ekP1afw1RT87wi6+Yx05e0tAQAB
com.ibm.WebSphere.ltpa.version=1.0
com.ibm.WebSphere.ltpa.Realm=rs617001.itso.ra1.ibm.com\:636
com.ibm.WebSphere.ltpa.PrivateKey=oxlHm1lro/BZoAOD1/WAPjCvRaxSr0cSBznq1PpCXRe3J
5+TcyeTrAQbOX6rGXLtBB1i/YKKPCQHSaV741Sak1j4iIaFK8UI0oeuU4u08ySyrbAIQaAJV+mUXUz
sxow592K899m4GkKK0JsV07dhnBWXAwtxHtoLZT7Y1iYaNiV4g4Y87JVhCaSJ/qWRogZ7Jn0Ih4PQXC
9nIib7mF8BMZBQzjyeDDSa+LCmFJ99osEDeh1oJ8cuUjVJz/61kn+zBtJ9NdEqh87aGFIAnbX1frq03
7r422Am5sv2YAD+qfPjr7MaTzWHYXNUTkgg9RH7SHeck/FYFFg3ZVqU+ZmMwIYmpsVecQ4BHTmLRSE
wg\=
com.ibm.WebSphere.CreationHost=M23CABYG
com.ibm.WebSphere.ltpa.3DESKey=+00oL0bpseEBN4PhW9pwy4Q07AgDoI5bp7ji+BjPSDM\=
```

14.3.2 Enabling SSO to use SSL in Domino

Follow the instructions explained in “Enabling Single Sign-On for Domino” on page 400, taking into account the following:

1. When you create the Directory Assistance document in the LDAP tab, specify the following values for the LDAP Configuration settings:
 - a. In the *Host Name* field, include the DNS host name for the IBM SecureWay Directory, for example: `rs617001.itso.ral.ibm.com`.
 - b. Enter a distinguished name in the *Username* field and a password in the *Password* field within the *Optional Authentication Credential* field. In our example, we created a new user in SecureWay Directory with the following distinguished name (DN): `cn=Domino Admin,ou=ITS0,o=Webbank`.
 - c. Fill out the *Base DN* field with the starting point for LDAP searches, for example `o=Webbank`. This field is required for the SecureWay Directory.
 - d. Select **Notes clients/Web Authentication** in the *Perform LDAP search for:* field.
 - e. Choose **SSL** in the *Channel Encryption* field to allow Domino Server to connect with the LDAP server using SSL.
 - f. Specify **636** in the *Port* field.
 - g. Choose **No** in the *Accept expired SSL certificates* field to enforce certificates' expirations dates.
 - h. Choose **Negotiated** in the *SSL protocol version* field to allow SSL to determine handshake and protocol.
 - i. Select **Enabled** in the *Verify server name with remote server's certificate* field, to require that the subject line of the remote server's certificate include the LDAP directory server host name.
 - j. Enter the maximum number of seconds before a search terminates in the *Timeout* field. The default is 60.
 - k. Enter the maximum number of entries a single search can return in the *Maximum number of entries returned* field. The default value is 100.

All these settings are illustrated in Figure 14-21.

DIRECTORY ASSISTANCE		
Basics	Rules	LDAP
LDAP Configuration		
Hostname:	rs617001.itso.ra1.ibm.com	
Optional Authentication Credential:		
Username:	cn=Domino Admin,ou=ITSO,o=webbank	
Password:	password	
Base DN for search:	o=webbank	
Perform LDAP search for:	<input checked="" type="checkbox"/> Notes Clients/Web Authentication <input type="checkbox"/> LDAP clients	
Channel encryption:	SSL	
Port:	636	
Accept expired SSL certificates:	No	
SSL protocol version:	Negotiated	
Verify server name with remote server's certificate:	Enabled	
Timeout:	60 seconds	
Maximum number of entries returned:	100	

Figure 14-21 SSL settings in the Directory Assistance document

2. Make sure, when importing WebSphere LTPA keys in the Web SSO Configuration document, to add a backslash (\) prior to the colon (:) before you save the document. The LDAP Realm field should have the following value:

```
rs617001.itso.ra1.ibm.com\:636
```
3. Make sure that the TCP/IP Port field specifies **Disabled** and the SSL Port field specifies **Enabled** in the Web tab (**Ports -> Internet Ports -> Web tab**) in the server document.

 Select **Yes** in the Name & Password field in the Authentication options section, as shown in Figure 14-22.
4. Make sure you have included the Domino certificate key file name in the SSL key file name field, as shown in Figure 14-22.

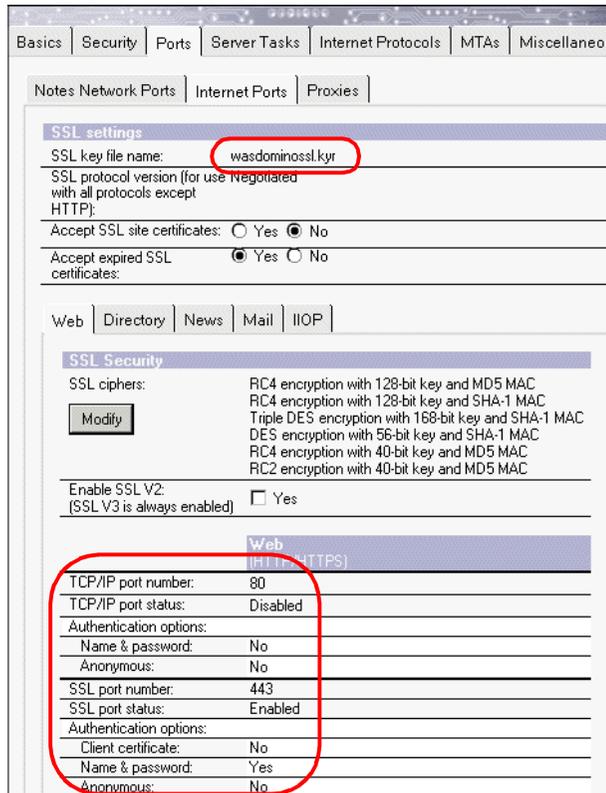


Figure 14-22 SSL settings in the Server Document

14.3.3 Testing SSO between Domino and WebSphere using SSL

To test SSO between Domino and WebSphere using SSL, repeat the same steps executed in “Testing Single Sign-On between Domino and WebSphere” on page 411, using **https** instead of http in the URLs, as shown for example in Figure 14-24 on page 423.

The first thing you will see when you specify Domino and WebSphere URLs is a security information box warning you that the certificate that the Web server presents to you is not known by your browser. Select **Yes** (see Figure 14-23 on page 423).

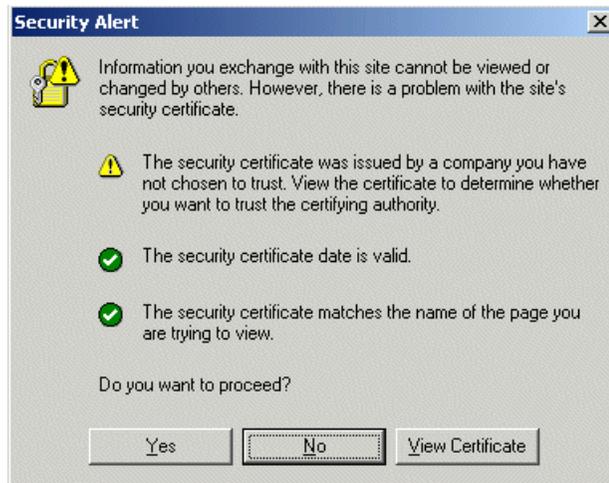


Figure 14-23 Security Alert Dialog box

The yellow lock appears at the bottom left of the Web browser window to indicate that you are using a secure connection, as shown in Figure 14-24.



Figure 14-24 Secure connection icon in the browser

Once you have logged into Domino or into WebSphere, click the following links to move between applications: **Please, send your comments (Secure)** (to go to Domino from WebSphere) or **Visit our secure site** (<https://www.itsoWebbank.com> (to go to WebSphere from Domino)).

14.4 WebSphere-Domino using Domino LDAP

We used the following product versions in the example documented below:

- ▶ Windows 2000 with Service Pack 2.
- ▶ 1 GB of RAM and 16 GB of hard disk space.
- ▶ IBM DB2 Universal Database V7.2.1, Enterprise Edition for Windows.
- ▶ IBM HTTP Server 1.3.19 for Windows as the Web server installed on the same machine as the WebSphere server.
- ▶ Lotus Domino Server R 5.06a.

- ▶ WebSphere Application Server Advanced Edition 4.01.
- ▶ Microsoft IE 5.5 for the Web user.

The example is shown in Figure 14-25.

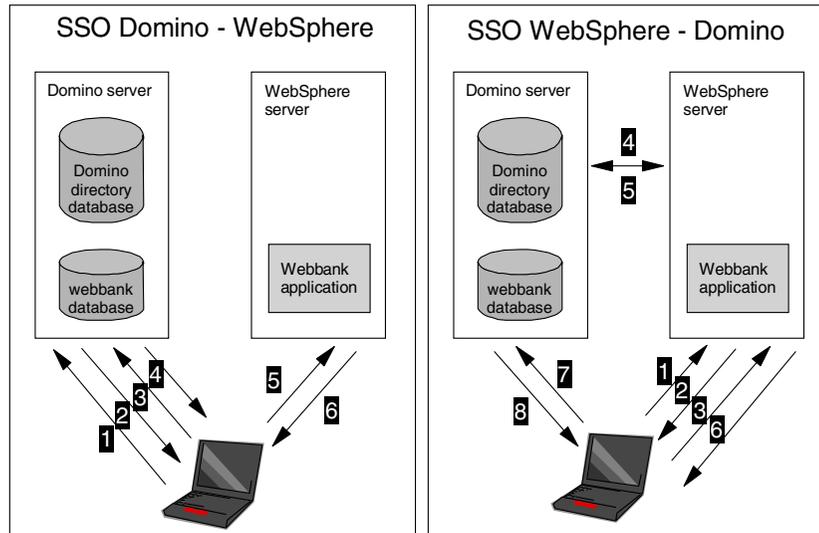


Figure 14-25 Sample Domino-WebSphere SSO using Domino Directory

WebSphere-Domino Single Sign-On

The following steps will describe the Single Sign-On process between Domino and WebSphere, when the user logs into WebSphere first.

1. A Web user submits a request to the Web server (HTTP Server) for a protected resource, to make a new bank transfer.
2. The Web server prompts the user for the authentication information.
3. The user responds by supplying the information (user name and password or certificate).
4. Then the Web server contacts the LTPA server (WebSphere), which connects with the Domino Directory to verify the authentication information.
5. If the information supplied for the user is correct, Domino responds to the server (WebSphere) with the validated information.
6. The server uses the returned values to check whether the user has access to the requested resource, then issues an LTPA token for the user. The Web server sends the token to the user as an HTTP cookie, which is stored in the user's browser and serves the requested resource (the webbank.html page in case of WebSphere).

7. Once the user is authenticated and the cookie is available, that user can request another protected resource from Domino or WebSphere.
8. Domino/WebSphere validate the token provided for the user and tell the Web server to send the requested resource to the browser, as long as the user has proper access to that resource, without prompting again with the challenge information.

Domino-WebSphere Single Sign-On

The followings steps will describe the Single Sign-On process between Domino and WebSphere, when the user logs into Domino first.

1. A Web user submits a request to the Web server (Domino) for a protected resource, to create a new Comment document in the Webbank Comment Application database.
2. Domino prompts the user for the authentication information.
3. The user responds by supplying the information (user name and password or certificate).
4. Domino then verifies the authentication information in the Domino directory, checks whether the user has rights to access to database and issues an LTPA token for the user as an HTTP cookie, which is stored in the user's browser. It then serves the requested resource (it opens the new Comment document).
5. Once the user is authenticated and the cookie is available, that user can request another protected resource from Domino/WebSphere.
6. Domino/WebSphere validate the token provided for the user and tell the Web server to send the requested resource to the browser, as long as the user has proper access to that resource, without prompting again with the challenge information.

The necessary steps to set up Single Sign-On between WebSphere and Domino involves:

- ▶ Installing and configuring software products and application examples
- ▶ Enabling Single Sign-On for the WebSphere Application Server
- ▶ Enabling Single Sign-On for the Domino Server

14.4.1 Installing and configuring software products and examples

For more details on how to install and configure the software products, refer to the installation manuals.

Make sure, before you begin to enable SSO, that the Webbank application has been installed in WebSphere and the Webbank Comment Application database has been created in Domino. For more details, refer to Chapter 5, “The sample used in this book” on page 73.

Directory structure

Once the Domino server is installed, we created the following Users and Groups in the Domino Directory.

Table 14-3 Domino Directory structure

User's Full Name	User's Short Name	Groups
Was Admin/ITSO	wasadmin	
Notes Administrator/ITSO	nadminis	Administrators
Marco Fuchs/ITSO	MarKo	Manager
James Roca/ITSO	James	Employee

We configured the Domino LDAP server task listening on port 390 and we did not allow anonymous LDAP connections. For more details about this feature, refer to Chapter 16, “IBM WebSphere Application Server and LDAP” on page 459.

14.4.2 Enabling Single Sign-On for WebSphere Application Server

Follow the instructions detailed in “Enabling Single Sign-On for WebSphere” on page 397.

Make sure to configure WebSphere to use Domino 5.0 as the directory type in the Security Center. For more details, refer to Chapter 16, “IBM WebSphere Application Server and LDAP” on page 459.

Before configuring Domino LDAP in the Security Center, make sure that the Domino server is running and the LDAP task is started, because the security server ID and security server password will be verified against Domino.

Important: Remember that the generation of the LTPA keys must be performed when the Domino LDAP server settings are configured; this guarantees that the LDAP host name and port are present in the exported file. Domino needs this information for the Web SSO Configuration Document creation process.

14.4.3 Enabling Single Sign-On for the Domino Server

The SSO setup in Domino, for our example, involves the following tasks:

- ▶ Setting up Domino for SSO.
- ▶ Implementing the security to the webbank.nsf database.

Setting up Domino for Single Sign-On

To set up SSO for the Domino Server, follow these steps:

1. Create a new Web SSO Configuration Document in the Domino Directory database. For more details, refer to “Enabling Single Sign-On for Domino” on page 405.
2. Enable the TCP/IP port status in the Web tab (**Ports -> Internet Ports -> Web** tab) and do not allow anonymous connections over TCP/IP by modifying the Domino Server Document.
3. Select a **Multi-Server** session in the Domino Web Engine tab (**Internet Protocols -> Domino Web Engine** tab) in the server document.
4. Select **More name variations with lower security** in the Web server authentication in the Security tab within the server document, to allow users to enter the following name formats in the name and password dialog box:
 - Last Name
 - First Name
 - Common Name
 - Full hierarchical name (canonical)
 - Full hierarchical name (abbreviated)
 - Short name
 - Alias name (a name listed in the User name field of the Person document, excluding the first name listed in the field)
 - Soundex number

14.4.4 Implementing security to the Webbank.nsf database

To implement the security into the Webbank Comments Application database, modify the Database ACL, and add the following Groups:

Table 14-4 Webbank Comments Database ACL Domino Users

People, Servers and Group	User Type	Access Level	Authorization
-Default-	Unspecified	No Access	None
LocalDomainServers	Server Group	Manager	Delete Documents
OtherDomainservers	ServerGroups	No access	None
Anonymous	Unspecified	No access	None
Administrators	People Group	Manager	Delete Documents
Employee	People Group	Editor	Delete Documents

All these settings are shown in Figure 14-26.

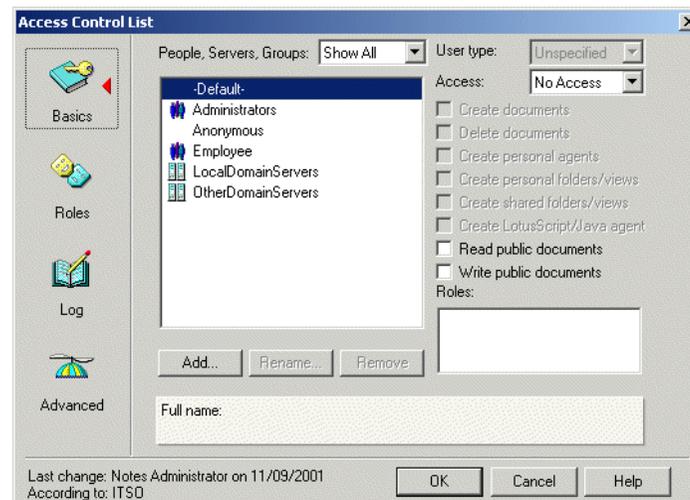


Figure 14-26 Webbank Comments Application ACL with Domino Users

Then set the *Maximum Internet name & password access* field to **Editor** under the *Advanced* section.

Testing Single Sign-On between Domino and WebSphere

We demonstrate how to use SSO between Domino and WebSphere in the following scenarios:

- Testing SSO between Domino and WebSphere
- Testing SSO between WebSphere and Domino

Testing SSO between Domino and WebSphere

The Web user wants to create a new Comment Document in the Webbank Comment application database, specifying the Webbank database URL in its browser:

```
http://m23x2640.itso.ra1.ibm.com/Webbank.nsf/Comments?OpenForm
```

The Domino Server will present a default server Login page, as shown in Figure 14-14 on page 412.

The Web user introduces his or her user name (Short Name/User ID) and password (Internet password) and clicks the **Login** button.

The Domino Server checks whether the user is registered in the Domino Directory database and verifies that the values included in the user name and password dialog box are the correct ones. Then it checks whether the user has access to the database and finds that that user has Editor access, since he or she belongs to the Employee group included in the ACL.

Once the user is authenticated, Domino will create the LTPA token, send it to the user as an HTTP cookie and open a New Comments document for the user, as shown in Figure 14-16 on page 415.

At this point, the user can click the **Submit** button to save the document, or return to the Webbank application home page by clicking the link: **Visit our Public site in <http://www.itsoWebbank.com>**. This link will connect the user to the WebSphere URL (in our case

<http://m23cabyg.itso.ra1.ibm.com:90/Webbank/Webbank.html>) and will open the Webbank.html page without prompting again for the user name and password, as shown in Figure 14-17 on page 416.

Note: The WebSphere Application Server and Domino server are running on the same machine, and both of them are running the HTTP Server. In order to avoid port conflicts, WebSphere was set to use port **90** for HTTP requests.

Testing SSO between WebSphere and Domino

In this case, the user wants to make a new bank transfer, specifying the Webbank application URL in its browser:

```
http://m23cabyg.itso.ra1.ibm.com:90/Webbank/Webbank.html
```

The HTTP Server prompts the user for the authentication information, as shown in Figure 14-19 on page 417.

The user responds to the challenge by supplying the information (user name and password) and clicks the **Login** button.

WebSphere will connect to Domino to verify the authentication information. If the information supplied for the user is correct, Domino responds to WebSphere with the validated information.

WebSphere uses the returned values to check whether the user has access to the requested resource (Webbank.html) and issues an LTPA token for the user. The Web server sends the token to the user as an HTTP cookie, as shown in Figure 14-15 on page 414 and opens the webbank.html page, as shown in Figure 14-17 on page 416.

At this point, the user can introduce the data for the transfer, then clicks the **Transfer** button or sends a new comment to the bank site by clicking the link **Please, send your comments (public)**. This link will connect to the Domino server URL (in our case

```
http://m23x2640.itso.ra1.ibm.com/Webbank.nsf/Comments?OpenForm)
```

 and will open the Webbank Comments Application database, as shown in Figure 14-16 on page 415, without prompting again for user name and password.

14.4.5 SSO WebSphere-Domino using Domino LDAP with SSL

In this case, we demonstrate the use of SSO in a secure environment using Domino as the user registry. For that purpose, we have secured the communications between:

- ▶ The Web Browser and the Domino Server.
- ▶ The Web Browser and the HTTP Server.
- ▶ The Web Server plug-in and the WebSphere Application Server.
- ▶ WebSphere and the Domino Server.

Before we explain how to configure SSO in a secure environment, make sure that the following steps have been taken:

1. Certificates have been issued to:
 - the HTTP Server
 - Domino
 - the Web Server plug-in
 - WebSphere
2. The IBM HTTP Server is configured to use SSL.
3. The Web Server plug-in is configured to use SSL.
4. WebSphere is configured to use SSL.
5. The Domino Web Server is configured to use SSL.
6. The Domino LDAP Server is configured to use SSL.

For more details of how to configure SSL and create the certificates, refer to “Configuring the Web Server to support HTTPS” on page 239.

To enable Single Sign-On in a secure environment, do the following:

- ▶ Enable SSO to use SSL in WebSphere.
- ▶ Enable SSO to use SSL in Domino.

Enabling SSO to use SSL in WebSphere

Follow the instructions detailed in “Enabling SSO to use SSL in WebSphere” on page 419. The WebSphere security settings in the Security Center are as shown in Figure 14-27 on page 433.

Enabling SSO to use SSL in Domino

Follow the instructions detailed in “Enabling Single Sign-On for the Domino Server” on page 427, taking into account the following:

1. Make sure, when importing WebSphere LTPA keys in the Web SSO Configuration Document, to add a backslash (\) prior to the colon (:) before you save the document. The LDAP Realm field should show something similar to the following value:

```
m23vnx78.itso.ra1.ibm.com\:636
```

2. Make sure that the SSL Port field is set to Enabled and the TCP/IP Port field specifies **Disabled** in the Web tab (**Ports -> Internet Ports -> Web** tab) within the server Document.

Select **Yes** in the Name & Password field in the Authentication options section, as shown in Figure 14-22.
3. Make sure you have included the Domino certificate key file name in the *SSL key file name* field, as shown in Figure 14-22.

Testing Single Sign-On between Domino and WebSphere using SSL and Domino LDAP

To test SSO between Domino and WebSphere using SSL, follow the instructions detailed in “Testing SSO between Domino and WebSphere using SSL” on page 422.

14.4.6 Troubleshooting Single Sign-On

The following section discusses how to troubleshoot problems when SSO is configured between Domino and WebSphere.

- ▶ If the *Creating Web SSO Configuration* document fails:
 - Make sure that you are using at least version R5.05 Notes client or Domino Administrator client when importing the LTPA keys.
 - The Notes or Domino Administrator’s client home server must be in the same domain as the participating SSO servers. Check that the entry in the *Home server* field in the client’s location document is pointing to a server in the same domain as the Domino SSO participating servers.
- ▶ If the loading of the Web SSO configuration on HTTP startup fails:
 - Check that there is only *one* Web SSO document in the Web Configuration view of the Domino Directory and the *\$WebSSOConfigs* hidden view. To do so, open the Domino Directory by holding the **Shift** and **Ctrl** keys down and double-clicking the icon. This will open the directory with all the hidden views. If more than one document is present, delete the wrong one or delete all the documents and recreate a single new one.
- ▶ If authentication fails:
 - Make sure that Domino and WebSphere have been configured for the same LDAP directory, as we explained in the steps above. Check that you can authenticate in WebSphere or in Domino separately before you test SSO.
 - Keep in mind that the LTPA cookie stores the *full Distinguished Name (DN) of the user*; if you are using SSO with WebSphere and Domino and

the Domino Directory as your LDAP server, flat names will not work for a user. Make sure you placed *only* the hierarchical name (for example: Victoria Amor/ITSO) in the User Name field for SSO to work.

- All the URLs must specify the full DNS server name, not the host name or IP address, because the DNS name is included in the cookie.
- Make sure, when configuring the LDAP server with a port, to edit the Web SSO Configuration document and to place a backslash (\) prior to the colon (:) in the LDAPRealm field (LDAP host name\:port) before saving the document.
- If you are using Domino Directory as your LDAP directory, include the full DNS name of the host computer on which Domino is installed in the *Fully qualified Internet Host name* field under the Basics tab of the server document, as shown in Figure 14-27.

The screenshot shows the 'Edit Server' window for 'Web...' with the title 'SERVER: SSODomino/ITSO'. The 'Basics' tab is selected. The 'Fully qualified Internet host name' field is circled in red and contains the value 'M23VNX78.itso.ral.ibm.com'. Other fields include 'Server name: SSODomino/ITSO', 'Domain name: ITSO', and 'Cluster name:'. The 'Optimize HTTP performance' section is set to 'Advanced (Custom Settings)'.

Basics	
Server name:	SSODomino/ITSO
Server title:	
Domain name:	ITSO
Fully qualified Internet host name:	M23VNX78.itso.ral.ibm.com
Cluster name:	
Directory Assistance database name:	
Directory Catalog database name on this server:	
Optimize HTTP performance based on the following primary activity:	Advanced (Custom Settings)

Figure 14-27 Fully Qualified Internet Host Name for the Domino Server

- If you are using clustered servers, they must include the host name with the full DNS server name in the *Fully qualified Internet Host name* field under the Basics tab of the server document for Domino Internet Cluster Manager (ICM) to redirect to cluster members using SSO.



Part 4

Appendixes



Problem determination

This chapter is provided as a supplement to the chapter on troubleshooting in the *WebSphere V4.0 Advanced Edition Handbook*. We specifically look at security-related problem determination techniques.

The topics discussed in this chapter include:

- ▶ The IBM HTTP Server (IHS)
- ▶ The WebSphere V4.0 Web Server plug-in
- ▶ The following aspects of the IBM WebSphere Application Server:
 - Enabling a security trace on the Administrative Server
 - Enabling a security trace on a managed Application Server
 - Enabling a Java Virtual Machine (JVM) trace argument
 - Enabling a Secure Association Service trace
 - The WebSphere Log Analyzer Tool
 - Example traces

15.1 The IBM HTTP Web Server

An important and often neglected aspect of managing any Web Server is problem determination. Such occurrences, while uncommon in a correctly specified configuration, are perhaps more likely to be encountered during the initial configuration and testing phases of deployment. Our experience tells us that among the top causes of problems are the failure to correctly validate an installation and the misinterpretation of a configuration parameter.

In the section that follows, we detail an approach that can be adopted for isolating problems associated with the IBM HTTP Server (IHS).

15.1.1 First steps

Many problems may be encountered within the system; possibly the IBM HTTP Server (IHS) fails to start, or there is a failure to serve a secure HTTPS request. Whatever the problem, it is always a nuisance, even for the most patient of administrators.

Proper administration and troubleshooting is all about trying to isolate the failing component, or eliminate the suspected offender. In either case, you should thoroughly check the following points before proceeding:

- ▶ Do we need clarification of the exact product version, fix level and platform environment?
- ▶ Is this an initial configuration? Has the configuration ever worked successfully?
- ▶ If this is not an initial configuration, what action was taken to potentially cause the problem? Was a parameter changed, a FixPack applied?
- ▶ Does the problem occur immediately upon starting the server?
- ▶ Alternatively, does the problem build up over time?
- ▶ Does the problem manifest itself when using an IBM sample configuration?
- ▶ Are any side effects, any CPU or memory problems encountered?
- ▶ Is the problem reproducible? Can you identify what action causes the problem?
- ▶ Is the problem reproducible on another system or platform?

Food for thought: how many System Administrators actually document the exact parameters of a configuration and the step-by-step procedures they took when installing the software product?

15.1.2 Problem determination

Determining the source of a problem associated with the IBM HTTP Server (IHS) can become a much easier task if you capture the correct data. Consider therefore changing the LogLevel directive from Warn (the default) to Debug. To achieve this, manually edit the httpd.conf file found under the IHS conf directory, using the editor of your choice:

```
..
ErrorLog /usr/HTTPServer/logs/error_log
LogLevel debug
..
```

If you have enabled a VirtualHost entry in your IHS configuration, possibly to listen on an SSL/HTTPS enabled port, you should also define a separate ErrorLog and TransferLog. This way, you can capture additional debugging information, specific to the concerned VirtualHost. As such, a completed VirtualHost stanza will look something like this:

```
..
<VirtualHost www.internetchaos.com:443>
ServerName www.internetchaos.com
ErrorLog logs/internetchaod443error_log
TransferLog logs/internetchaos443access_log
SSLEnable
SSLServerCert WebServer
SSLClientAuth required
SSLCipherSpec 33
SSLCipherSpec 36
</VirtualHost>
..
```

Be sure to restart the IHS after making any modifications so that the changes will be included in the runtime.

15.1.3 Web Server trace example

One such occasion that might merit changing the IHS LogLevel directive to Debug is when problems with client side certificates are encountered. Here, as demonstrated in Example 15-1, the enhanced LogLevel directive can be used to capture the client certificate submitted to the Web Server.

Example 15-1 Client Side Certificate submission

```
...
[info] Cert Body:
MIICXzCCAciGAWIBAgIDUdboMAOGCSqGSIb3DQEBBAUAMIGHMQswCQYDVQQGEwJaQTEiMCAGA1UECBM
ZRk9SIFRFU1RJTkcgUFVSUE9TRVMgT05MWTEdMBSGA1UEChMUVGhhd3R1IEN1cnRpZm1jYXRpb24x
FzAVBgNVBAsTD1RFU1QgVEVTVCBURVNUMRwwGgYDVQQDExNUaGF3dGUgVGZzdCBDQSB290MB4XDTAxM
```

```

TEwNzAxMjY1MFoXDTAxMTEyODAxMjY1MFowYTETMBEGCgmSJomT8ixkARkTA2NvbTEdMBsGCgmSJomT8ixkARkTDWludGVybmVOY2hhb3MxCzAJBgNVBAsTAnVrMQ4wDAYDVQQLEwV1c2VyczEOMAwGA1UEAxMFYW1vcnYwgZ8wDQYJKoZIhvcNAQEBBQADgYOAMIGJAoGBALGHqILI6oadtio9rFsZ21aKzSIy1xc87p
yaCNZDtm5ZyOrjWZ9pc3BP1SwaQLa+s9rnozJ2I XuHzKESN8DSmcmRGE+SL+6vzgAH/MDKgooRVyw
Y10Zok6GiW6M4xjxmEWvEpvo0BYzUcXTJsCusmwQQ2aFfia9uknHgmrRmmZAgMBAAEwDQYJKoZIhvcN
AQEEBQADgYEAGXyeBDKLB8cJKeD1N1VU3qyCffj5dWoOYPmfIRoRqnQ3rqPmFJhd/HrWJMIB645isEX
2LDSsqOCZSTbbKJ876ecsri/oQ0bnyasWP6cHWFm6VoxmGwcXw6FMf1oXQWT5ACqaBoTWZHKR63Qakz
Ko87S+QMkwEqFSpH03d1qSn5E=RmmZAgMB ^EEö
[info] Cert Body Len: 816
[info] Serial Number: 51:d6:e8
[info] Distinguished name CN=amorv,OU=users,OU=uk,DC=internetchaos,DC=com
[info] Common Name: amorv
[info] Organization Unit: uk, users
[info] Issuer's Distinguished Name: CN=Thawte Test CA Root,OU=TEST TEST
TEST,0=Thawte Certification,ST=FOR TESTING PURPOSES ONLY,C=ZA
[info] Issuer's Common Name: Thawte Test CA Root
[info] Issuer's State or Province: FOR TESTING PURPOSES ONLY
[info] Issuer's Country: ZA
[info] Issuer's Organization: Thawte Certification
[info] Issuer's Organization Unit: TEST TEST TEST
...

```

Remember that client side certificates are only requested from client Web browsers if the SSLClientAuth directive is specified in the IHS httpd.conf file.

15.2 The WebSphere Web Server plug-in

After an HTTP request has successfully been handled by the selected Web Server of your choice, it is up to the WebSphere plug-in resident in the Web Server to determine if the resource requested is one served by WebSphere. If it is, knowing how to properly debug the plug-in when problems occur can greatly facilitate a speedy resolution.

Management of the WebSphere Web Server plug-in has been greatly simplified, with the parameters now being specified in a single plugin-cfg.xml file.

Note: The separate vhost.properties, queues.properties and rules.properties files that shipped with previous WebSphere versions are now absent.

15.2.1 First steps

When dealing with the IHS, we introduced the concept of an initial set of questions to ask. Diagnosing problems with the WebSphere Web Server plug-in is no different. We revisit the questions that should be explored prior to getting too involved in any low-level debugging. It is important to appreciate that knowing the answers to the questions, will greatly facilitate a speedy resolution if you decide to engage IBM Support.

- ▶ Do we need clarification of the exact product version, fix level and platform environment?
- ▶ Is this an initial configuration? Has the configuration ever worked successfully?
- ▶ If this is not an initial configuration, what action was taken to potentially cause the problem? Was a parameter changed, a FixPack applied?
- ▶ Does the problem occur immediately upon starting the server?
- ▶ Alternatively, does the problem build up over time?
- ▶ Does the problem manifest itself when using an IBM sample configuration?
- ▶ Are any side effects, any CPU or memory problems encountered?
- ▶ Is the problem reproducible? Can you identify what action causes the problem?
- ▶ Is the problem reproducible on another system or platform?

15.2.2 Problem determination

When problems are encountered with the WebSphere Web Server plug-in, enabling tracing can quickly help to isolate the failing component or misconfiguration. As such, the task is simply accomplished by changing the LogLevel attribute in the plugin-cfg.xml file from Error (the default) to Trace. This is shown below:

```
<?xml version="1.0"?>
<Config>
  <Log LogLevel="Trace" Name="/usr/WebSphere/AppServer/logs/native.log"/>
  ..
```

After modifying the LogLevel attribute, ensure that you restart the IHS or the Web Server of your choice, as this will ensure that the increased trace specification is immediately implemented. You should consider clearing the log files prior to capturing the problem that has occurred. Otherwise, you will be faced with an excessive amount of data to analyze. Remember that although the problem may

appear obvious to you, you can greatly improve the response from your local support team if the immediate issue is detailed and void of any ambiguity. For example, it has not been uncommon for customers to forward in a week's worth of logs to IBM Support.

15.2.3 Web Server plug-in trace example

Changing the WebSphere Web Server plug-in LogLevel attribute to Trace can be beneficial when problems are experienced with SSL handshaking between the plug-in and the corresponding WebSphere Web Container.

In such an event, it may be possible to verify that the embedded HTTP Server affiliated with the WebSphere Web Container is functioning correctly by pointing a client Web browser directly to the embedded HTTP Server IP address and secure listening port number. If the embedded HTTP Server has been configured with SSL client authentication enabled, the browser will be prompted to submit a client side certificate. Otherwise, the resource requested by the Web browser will only be displayed if the embedded HTTP Server is functioning correctly.

Example 15-2 illustrates the type of error message that can be captured in the plug-in native.log file when SSL handshaking fails between the plug-in and a WebSphere Web Container.

Example 15-2 native.log

```
...
ERROR: lib_stream: openStream: Failed in r_gsk_secure_soc_init:
      GSK_ERROR_BAD_CERT(gsk rc = 414)
ERROR: ws_common: websphereGetStream: Could not open stream
ERROR: ws_common: websphereExecute: Failed to create the stream
ERROR: ws_common: websphereHandleRequest: Failed to execute the transaction to
      'Default Server'; will try another one
ERROR: ws_common: websphereHandleRequest: Failed to find an app server to
      handle this request
...
```

Generally, it is always worth consulting the logs of the associated SSL peer when problems of an SSL handshaking nature occur. On this occasion, it is the IBM WebSphere Application Server stdout and stderr log files, responsible for the peer Web Container, that may offer further clues as to the origin of the problem. As such, one permissible error message that might be seen in the peer Application Server stdout log file is shown in Example 15-3.

Example 15-3 Default_Server_stdout.log

```
2333b974 HttpConnectio W bad certificate; perhaps the server's SSL security
      level is higher than the client can perform
```

A further clue is also revealed by the IBM WebSphere Application Server stdout log file and is shown in Example 15-4. Here, the embedded HTTP Server, or rather, the secure HTTPS transport protocol associated with the embedded HTTP Server, is seen to successfully start listening on TCP port 9080. Those familiar with WebSphere V4.0 will know that the embedded HTTP Server will fail to start if the certificate database/keyfile referenced by the Web Container is incorrectly configured.

Example 15-4 Default_Server_stdout.log

```
HttpTransport A SRVE0171I: Transport https is listening on port 9,080.
```

The message, however, does not provide any guarantee that the certificate keys present in the Web Container certificate database/keyfile are specified or correctly configured.

In this case, the problem is most likely to be caused by one of the following:

1. The SSL peer certificate databases/keyfiles do not contain the correct certificate keys. Check, if using private/public self-signed certificate key pairs, that the public certificate keys are exchanged successfully between the peers. Alternatively, if using a third-party Certificate Authority (CA), check that the public certificate key of the CA is installed as a Trusted Signer in the adjacent certificate database of the SSL peer.
2. The private/public certificates do not contain all the information they need. As such, check each certificate Common Name (CN) and verify that associated fields are completed correctly.
3. The certificate label name or alias set when importing the public certificate key from either adjacent peer contains one or more spaces. Delete the exchanged public keys and re-add the certificate keys under the respective ikeyman Signer Certificates menu, ensuring that the new label name contains no spaces.
4. The encryption algorithms or cipher strength supported between the adjacent SSL peers is not compatible. It is permissible to restrict the SSL supported cipher strength at the WebSphere Web Container level; check this setting.

15.3 The IBM WebSphere Application Server

Dealing with IBM WebSphere Application Server problems can at first seem a daunting prospect for even the most accomplished administrator. However, with a little knowledge and direction, you can quickly master a situation, identifying and rectifying the problem successfully.

In this section, we endeavour to share with you some of the techniques commonly used and endorsed by the IBM WebSphere Support Team. As the focus is primarily on WebSphere V4.0 security issues, you should consider reviewing the chapter concerning troubleshooting in the *WebSphere V4.0 Advanced Edition Handbook* (SG24-6176) for a more holistic approach to problem determination, if needed.

15.3.1 First steps

Understanding the components involved with WebSphere security will greatly enhance your diagnostic skills. Figure 15-1 highlights the main Java classes responsible for implementing security.

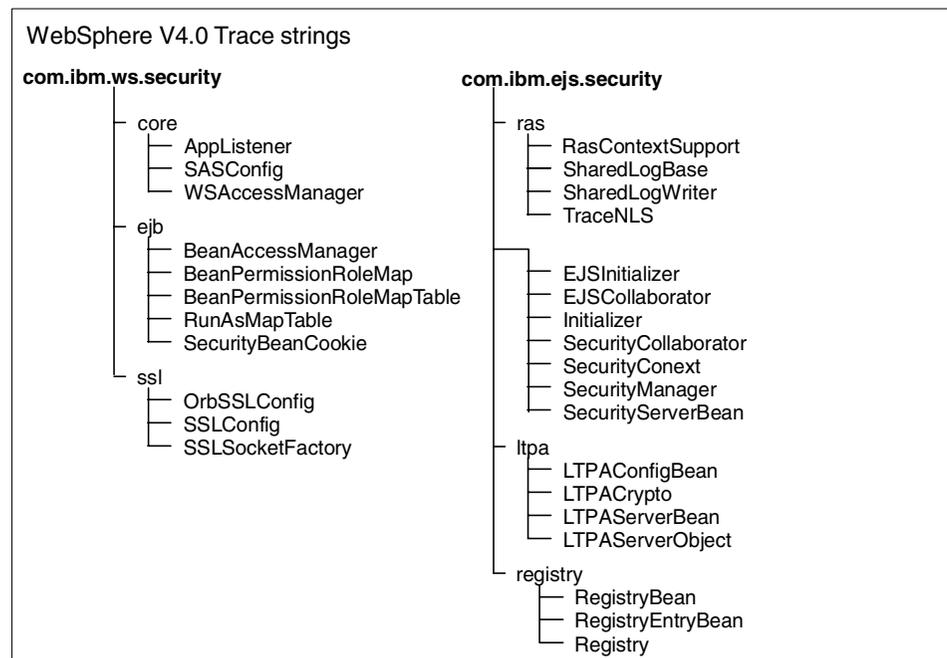


Figure 15-1 WebSphere V4.0 trace strings

By virtue of the tasks these classes perform, we can readily enable tracing on any of the classes to home in on a suspected problem. We recognize that many administrators may not be familiar with the exact workings of each class and, for this reason, we suggest that tracing be implemented on a more general level.

For example, failure to bind to an LDAP Directory server may throw an exception in the `LdapRegistryImplementation`. However, it is not immediately evident which class or component contains this implementation. In this case, enabling tracing at the `com.ibm.ejs.security` level will capture a complete yet exhaustive trace of every component class in the `com.ibm.ejs.security` package. The trace specification can then be further refined on subsequent operations.

There are, of course, many other allowable possible permutations of the trace settings. As such, the starting point, when in doubt for diagnosing any security-related problem, can be the following strings:

```
com.ibm.ws.security.*=all=enabled  
com.ibm.ejs.security.*=all=enabled
```

Later on in this section, we will demonstrate the exact use of these trace strings.

Needless to say, problem determination also revolves around eliminating potential suspects. In this case, as already introduced with the debugging of the IHS and the WebSphere Web Server plug-in, you should be in a position to clarify the following points:

- ▶ Do we need clarification of the exact product version, fix level and platform environment?
- ▶ Is this an initial configuration? Has the configuration ever worked successfully?
- ▶ If this is not an initial configuration, what action was taken to potentially cause the problem? Was a parameter changed, a FixPack applied, new code deployed?
- ▶ Does the problem occur immediately upon starting the IBM WebSphere Administrative Server?
- ▶ Does the problem occur immediately upon starting a managed Application Server?
- ▶ Alternatively, does the problem build up over time?
- ▶ Does the problem manifest itself when using an IBM sample configuration?
- ▶ Are any side effects, any CPU or memory problems encountered?
- ▶ Is there any correlation to the load experienced by the Application Server?
- ▶ Is the problem reproducible? Can you identify what action causes the problem?
- ▶ Is the problem reproducible on another system or platform?

15.3.2 Problem determination

WebSphere readily supports the ability to enable security level tracing on both the Administration Server and on any individually configured Application Server. Each approach, however, is slightly different, as the Administrative Server is typically traced by setting the trace strings in the admin.config file.

It is also worth remembering that the Administrative Server performs such tasks as authenticating and authorizing users on behalf of the managed Application Servers. As such, the SecurityCollaborator, RegistryBean, RegistryImplementation and LdapRegistryImplementation are functions of the Administrative Server and appear transparent to a managed Application Server. However, as will become apparent, such functions are critical to the viability of any managed Application Server.

The Administration Server

To facilitate a comprehensive security trace of the Administration Server, edit the admin.conf file found in the WebSphere bin directory and add the following two lines:

```
com.ibm.ejs.sm.adminServer.traceString=com.ibm.ws.security.*=all=enabled:co
m.ibm.ejs.security.*=all=enabled
com.ibm.ejs.sm.adminServer.traceOutput=/usr/WebSphere/AppServer/logs/admin.
trace
```

By default, WebSphere ships with the traceString and traceOutput directives hashed out. Enable tracing by removing the preceding hash (#), specifying the trace string of your choice as well as the output file.

Managed Application Server

This section is dedicated exclusively to enabling tracing on a WebSphere managed Application Server.

1. Start the WebSphere Administrative Console and select the managed Application Server that you wish to enable tracing on from the topology management tree. Proceed by clicking the right mouse button and then selecting the resulting **Properties** tab.
2. You will be presented with the window shown in Figure 15-2. Make sure that the Services tab is currently in focus and select the **Trace Service**. Click the **Edit Properties** button to launch the Trace specification window.

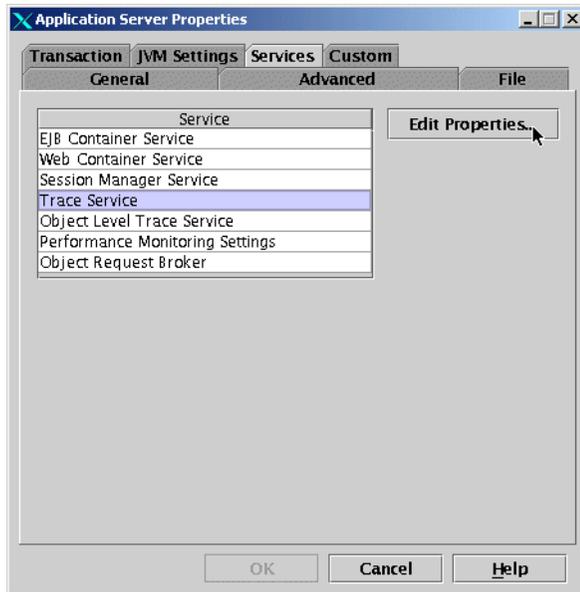


Figure 15-2 Application Server properties

3. If you know the exact trace requirements that you wish to implement, you can simply type the string in the Trace Specification field. Alternatively, if you click the button at the right hand side of the Trace Specification field, you will launch the hierarchal trace tree. This is discussed in detail in the *WebSphere V4.0 Handbook*, SG24-6176.

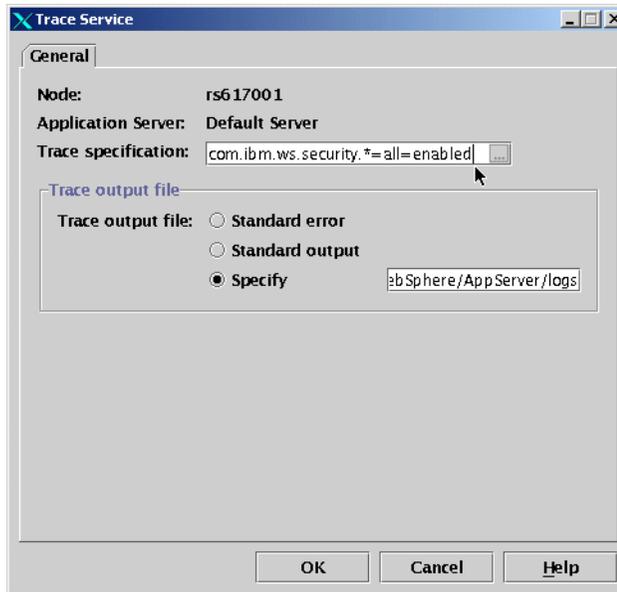


Figure 15-3 Application Server's properties

- To capture a comprehensive security trace on an Application Server, edit the following two settings:

For the Trace specification, set the following value:

```
com.ibm.ejs.security.*=all=enabled:com.ibm.ws.security.*=all=enabled
```

For the Trace output file, fill in the fully qualified filename:

```
/usr/WebSphere/AppServer/logs/appsecuritytrace.log
```

15.3.3 JVM trace arguments

In addition to modifying the trace specification settings, it is also possible to pass a trace argument directly to the Java Virtual Machine (JVM) associated with each Application Server. Again, there are two slightly different approaches for achieving this, depending on whether you are passing an argument to a JVM associated with the Administrative Server, or to a JVM affiliated with a managed Application Server.

In the case of the Administrative Server, this is achieved by editing the `admin.config` file in the WebSphere bin directory and specifying the option you wish to pass as an Administrative Server JVM argument.

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-Djavax.net.debug=true ....
```

In contrast, if you choose to pass an argument to the JVM associated with a managed Application Server, you will most likely perform the task using the Administration Client console.

In this case, open the Application Server **Properties** window of your chosen managed Application Server and expand the **JVM Settings** to reveal the Advanced JVM Settings window. You can then, as shown below in Figure 15-4, specify the option you wish to pass to the JVM as a command line argument.

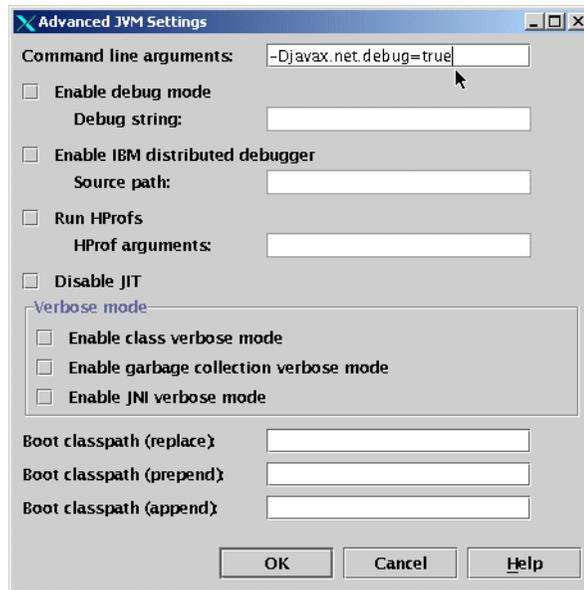


Figure 15-4 Specifying the option for the JVM

After clicking **OK**, you will need to restart your managed Application Server to ensure that the modification is included in the next runtime.

15.3.4 The Secure Association Service (SAS)

The final important trace that you can use in terms of security is a trace on the Secure Association Service. It can be implemented for both the server and client.

In the `sas.server.props` file, set the following debugging directives to get a detailed output of security activities.

```
com.ibm.CORBA.securityErrorsOutputMode=both
com.ibm.CORBA.securityTraceOutputMode=file
com.ibm.CORBA.securityTraceOutput=/opt/WebSphere_4.0.1_AE/AppServer/logs/sa
s_server.log
com.ibm.CORBA.securityTraceLevel=intermediate
```

```
com.ibm.CORBA.securityActivityOutputMode=file
com.ibm.CORBA.securityExceptionsOutputMode=file
com.ibm.CORBA.securityDebug=true
```

Example 15-5 is an excerpt from the *tracefile* where a detailed security log was written.

Example 15-5 tracefile excerpt

```
JSAS0240E: Login failed. Verify the userid/password is correct. Check the
properties file to ensure the login source is valid. If this error occurs on
the server, check the server properties to ensure the principalName has a valid
realm and userid.
```

```
SECJ0131E: Authentication failed. Unable to get the mapped credential for
SecOwnCredentials.
```

```
SECJ0007E: Error during security initialization
```

```
SECJ0007E: Error during security initialization
```

```
WSVR0009E: Error occurred during startup
```

For more information about SAS, refer to Chapter 8, “Securing J2EE clients” on page 155.

15.3.5 Post-analysis using the Log Analyzer

Learn to take full advantage of the Log Analyzer tool, as it is often neglected. IBM is also actively investing in maintaining a downloadable symptom database. This is projected to grow so as to meet the needs of future administrators.

One potential use of the Log Analyzer tool is to determine the exact class of a failing exception, or the class responsible for executing an implementation when a failure occurs.

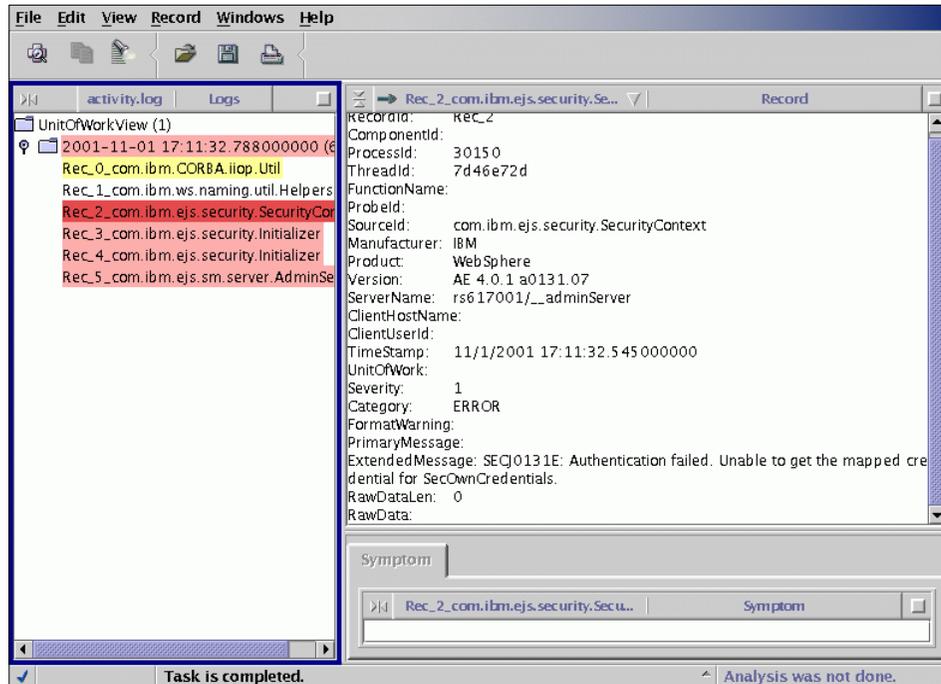


Figure 15-5 WebSphere Log Analyzer tool

One such example is illustrated above in Figure 15-5. Here, a problem is encountered when implementing the `com.ibm.ejs.security.SecurityContext` class, as reported by the `SourceId` field. The `ExtendedMessage` field offers a further explanation of the problem.

From the details given above, we are now in a position to trace the problem using the following trace string:

```
com.ibm.ejs.security.SecurityContext=all=enabled
```

However, this trace string may be too limited for the problem encountered. In this case, a more comprehensive analysis can be performed using the following string:

```
com.ibm.ejs.security.*=all=enabled
```

15.3.6 IBM WebSphere Application Server trace example

So far in this section, we have discussed the intricacies of enabling tracing on both the WebSphere Administration Server and any given managed Application Server.

By way of introducing the trace functionality offered by WebSphere, we first examine the output from a tracefile log example indicative of a successful startup of the IBM WebSphere Application Server; this is shown below in Example 15-6.

Example 15-6 tracefile log output

```

7d5ac167 Server      U Version : 4.0.1
7d5ac167 Server      U Edition: Advanced Edition for Multiplatforms
7d5ac167 Server      U Build date: Thu Aug 09 00:00:00 BST 2000
7d5ac167 Server      U Build number: a0131.07
7d5ac167 ORBRas      W com.ibm.CORBA.iiop.Util Util P=649553:0=0:CT
                        JORB0012: Pass by reference has been set to:true
                        (NoLocalCopies = true)
7d5ac167 DrAdminServer I WSVR0053I: DrAdmin available on port 63104
7d5ac167 AdminServer  I ADMS0008I: Initializing WebSphere Administration
                        server
7d5ac167 ResourceBinde I WSVR0049I: Binding SM_DATASOURCE as jdbc/SM_Datasource
7d5ac167 EJBEngine    I WSVR0037I: Starting EJB jar: Name Service
7d5ac167 EJBEngine    I WSVR0037I: Starting EJB jar: Repository
7d5ac167 EJBEngine    I WSVR0037I: Starting EJB jar: Tasks
7d5ac167 Server      A WSVR0023I: Server __adminServer open for e-business
19b90167 ActiveServerP A ADMS0008I: Starting server: Default Server
19b90167 ActiveServerP A ADMS0032I: Started server: Default Server (pid 27178)

```

As expected, the WebSphere Administrative Server is seen to open for e-business successfully, with the Default managed Application Server subsequently starting up assuming the process ID (PID) 27178. It is also noted that the Object Request Broker (ORB) associated with the Administrative Server is invoked with the NoLocalCopies attribute set to True. This is intentional and a result of having the `-Dcom.ibm.CORBA.iiop.noLocalCopies=true` attribute set in the `admin.config` file as an argument passed to the Administrative Server Java Virtual Machine (JVM).

In contrast, Figure 15-7 documents a failure encountered when starting the IBM WebSphere Application Server. Looking more closely at the tracefile, we know that everything appears to be starting up smoothly until we encounter the JSAS0240E: Login failure error. Under normal circumstances, we would expect to see the Administrative Server opening for e-business after the WSVR0036I: Starting EJB jar: Tasks routine.

Example 15-7 tracefile excerpt

```

7d46e72d Server      U Version : 4.0.1
7d46e72d Server      U Edition: Advanced Edition for Multiplatforms
7d46e72d Server      U Build date: Thu Aug 09 00:00:00 EDT 2001
7d46e72d Server      U Build number: a0131.07
7d46e72d ORBRas      W com.ibm.CORBA.iiop.Util Util P=649553:0=0:CT
                        JORB0012: Pass by reference has been set to:true
                        (NoLocalCopies = true)

```

```

7d46e72d DrAdminServer I WSVR0053I: DrAdmin available on port 41270
7d46e72d AdminServer   I ADMS0008I: Initializing WebSphere Administration
                    server
7d46e72d ResourceBinde I WSVR0049I: Binding SM_DATASOURCE as jdbc/SM_Datasource
7d46e72d EJBEngine     I WSVR0037I: Starting EJB jar: Name Service
7d46e72d EJBEngine     I WSVR0037I: Starting EJB jar: Repository
7d46e72d EJBEngine     I WSVR0037I: Starting EJB jar: Tasks
7f1ace51 SystemOut    U JSAS0240E: Login failed. Verify the userid / password
                    is correct. Check the properties file to ensure the
                    login source is valid. If this error occurs on the
                    server, check the server properties to ensure the
                    principalName has a valid realm and userid.
7d46e72d SecurityConte X SECJ0131E: Authentication failed. Unable to get the
                    mapped credential for SecOwnCredentials.
7d46e72d Initializer   X SECJ0007E: Error during security initialization
7d46e72d Initializer   X SECJ0007E: Error during security initialization
7d46e72d AdminServer   X WSVR0009E: Error occurred during startup

```

Those familiar with WebSphere will know that when the Application Server is first started up with Global Security enabled, the Administrative Server authenticates itself and assumes the identity defined by the Security Server ID setting. In the case of using the LTPA authentication mechanism, a request is sent to the remotely configured LDAP Directory Server.

At this point, the keyword SecOwnCredentials (underlined) offers a further clue that it is indeed the WebSphere Security Server ID identity that is failing to authenticate against the remote LDAP Directory Server.

Analyzing the accompanying *activity.log* with the Log Analyzer tool shows a similar set of entries to what was previously determined from the tracefile. As such, the `SourceId` and `ExtendedMessage` entries recorded are as follows:

```

SourceId:com.ibm.ejs.security.SecurityContext
ExtendedMessage: SECJ0131E: Authentication failed. Unable to get the mapped
credential for SecOwnCredentials.

```

```

SourceId:com.ibm.ejs.security.Initializer
ExtendedMessage: SECJ0007E: Error during security initialization
SourceId:com.ibm.ejs.security.Initializer
ExtendedMessage: SECJ0007E: Error during security initialization

```

```

SourceId:com.ibm.ejs.sm.server.AdminServer
ExtendedMessage: WSVR0009E: Error occurred during startup
java.lang.RuntimeException: Authentication Failed

```

```

SourceId:com.ibm.ejs.sm.server.AdminServer
ExtendedMessage: WSVR0067E: Failed to initialize WebSphere Administration
server

```

Potentially, the next step in resolving the issue could involve enabling a WebSphere trace.

It is here that the Log Analyzer output identifying the *SourceID* is beneficial, as it corresponds directly with the trace specification that should be used to capture the failure. As such, the Log Analyzer SourceID identifies the following implementation class for the problem experienced in Example 15-7 on page 452:

```
com.ibm.ejs.security.SecurityContext
```

Note: The tracefile log only reports the abbreviated *SecurityConte* when the problem is experienced. The full Java package name is not detailed, as it is with the Log Analyzer tool.

Also recall from Section 15.3.5, “Post-analysis using the Log Analyzer” on page 450 that although the Log Analyzer SourceID is valid, the trace specification may be too limited for the problem experienced. In this case, by forfeiting the SecurityContext object class, a more comprehensive analysis can be performed with the following specification:

```
com.ibm.ejs.security.*=all=enabled
```

To this end, and to facilitate the required WebSphere trace, the following entries need to be placed in the admin.conf file found in the WebSphere bin directory:

```
com.ibm.ejs.sm.adminServer.traceString=com.ibm.ejs.security.*=all=enabled
com.ibm.ejs.sm.adminServer.traceOutput=/usr/WebSphere/AppServer/logs/admin.
trace
```

You must then proceed to restart WebSphere for the trace specification to be included in the next runtime.

By way of acknowledging that our problem is most likely related to a malfunctioning remote LDAP Directory Server, we next explore some typical security traces, looking specifically at the function of LDAP authentication.

Example 15-8 illustrates a successful bind and authentication against a remote IBM SecureWay LDAP Directory Server. In itself, the action is the correct sequence employed by WebSphere upon initial startup, when Global Security is enabled.

The trace starts by capturing the authentication of the user *websphere*, as set according to WebSphere Security Server ID identity. From the filter option, it is apparent that WebSphere is attempting to authenticate the user against the *uid* field in the remote SecureWay LDAP Directory. The user happened to be stored in the *ePerson* objectclass type. The *getRootDSE* sequence is a special function that actually passes information describing the LDAP Directory to the requester, in this case WebSphere.

Note: the direction of the < > (chevrons) determines the flow of the requested information.

Example 15-8 Successful LDAP authentication

```
7d5ac167 LdapRegistryI D Authenticating websphere
7d5ac167 LdapRegistryI D Searching for users
7d5ac167 LdapRegistryI > getUsers websphere
7d5ac167 LdapRegistryI D filter = (&(uid=websphere)(objectclass=ePerson))
7d5ac167 LdapRegistryI > getRootDSE
7d5ac167 LdapRegistryI < getRootDSE
7d5ac167 LdapRegistryI < getUsers
7d5ac167 LdapRegistryI D Found user cn=websphere,ou=authenticated,
                        ou=uk,dc=internetchaos,dc=com
7d5ac167 LdapRegistryI D Authenticated with cn=websphere,ou=authenticated,
                        ou=uk,dc=internetchaos,dc=com
7d5ac167 LdapRegistryI > getGroupsForUser
7d5ac167 LdapRegistryI D filter = (|(&(objectclass=groupofnames)(member=cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com))
                        (&(objectclass=groupofuniquenames)(uniquemember=cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com)))
7d5ac167 LdapRegistryI > getRootDSE
7d5ac167 LdapRegistryI < getRootDSE
7d5ac167 LdapRegistryI < getGroupsForUser
7d5ac167 LdapRegistryI > getUserDisplayName
7d5ac167 LdapRegistryI > getRootDSE
7d5ac167 LdapRegistryI < getRootDSE
7d5ac167 LdapRegistryI D securityName =cn=websphere,ou=authenticated,
                        ou=uk,dc=internetchaos,dc=com
7d5ac167 LdapRegistryI D attributes ={uid=uid: websphere, objectclass=objectclasses: top, organizationalPerson, person, ePerson, inetOrgPerson, cn=cn: websphere}
7d5ac167 LdapRegistryI D userName = websphere
7d5ac167 LdapRegistryI < getUserDisplayName
7d5ac167 LdapRegistryI < authenticate
7d5ac167 SecurityColla < postInvoke
7d5ac167 Initializer D Authorization table found for admin application,
                        Adding serverId to all roles serverId = websphere
                        Accessid = user:rs617001.itso.ral.ibm.com:636/cn=websphere,ou=authenticated,ou=uk,dc=internetchaos,dc=com
7d5ac167 Initializer D Added ServerId for role AdminRole
7d5ac167 Initializer < bindServerIdToAdminApp
7d5ac167 SecurityColla > enableSecurity
7d5ac167 SecurityColla < enableSecurity
7d5ac167 Initializer < initialize
7d5ac167 Server A WSVR0023I: Server __adminServer open for e-business
```

The *userName* attribute is *websphere* and corresponds to the Bind Distinguished Name as set under the WebSphere Global Security settings. It is this identity that WebSphere uses to bind to the remote LDAP Directory and to perform user authentication. In Example 15-8, both the WebSphere Security Server ID identity and the Bind Distinguished Name are the same, with both attributes defined accordingly in the configuration.

Finally, thread *7d5ac167* invokes the *Initializer* class and we see the *serverId* Security Server ID identity *websphere* being added to the *AdminRole*, before the Admin Server opens for e-business.

A contrasting trace is shown in Example 15-9 below, for WebSphere authenticating against a remote Domino LDAP Directory Server. There are some subtle differences with the filtering mechanism used in the LDAP bind, but the sequence is more or less the same. One exception to note is that WebSphere is connecting anonymously to the remote Domino LDAP Directory Server. This is determined by the *userName* being reported as *<null>*. We strongly advise against connecting anonymously to a remote LDAP Directory Server.

Example 15-9 Anonymous Bind to Domino LDAP Server

```

31f2a7 LdapRegistryI D Authenticating wasadmin
31f2a7 LdapRegistryI D Searching for users
31f2a7 LdapRegistryI > getUsers wasadmin
31f2a7 LdapRegistryI D filter = (&(uid=wasadmin)(objectclass=dominoPerson))
31f2a7 LdapRegistryI > getRootDSE
31f2a7 LdapRegistryI < getRootDSE
31f2a7 LdapRegistryI < getUsers
31f2a7 LdapRegistryI D Found user CN=wasadmin
31f2a7 LdapRegistryI D Authenticated with CN=wasadmin
31f2a7 LdapRegistryI > getGroupsForUser
31f2a7 LdapRegistryI D filter =( | (&(objectclass=dominogroup)
(member=CN=wasadmin)))
31f2a7 LdapRegistryI > getRootDSE
31f2a7 LdapRegistryI < getRootDSE
31f2a7 LdapRegistryI D securityName =CN=wasadmin
31f2a7 LdapRegistryI D attributes ={objectclass=objectclass: top, person,
organizationalPerson, inetOrgPerson, dominoPerson,
cn=cn: wasadmin}
31f2a7 LdapRegistryI D userName =<null>
31f2a7 LdapRegistryI < getUserDisplayName
31f2a7 LdapRegistryI < authenticate
31f2a7 SecurityColla < postInvoke
31f2a7 Initializer D Authorization table found for admin application, Adding
serverId to all roles serverId = wasadmin Accessid =
user:rs617001.itso.ra1.ibm.com:389/CN=wasadmin
31f2a7 Initializer D Added ServerId for role AdminRole
31f2a7 Initializer < bindServerIdToAdminApp
31f2a7 SecurityColla > enableSecurity

```

```
31f2a7 SecurityColla < enableSecurity
31f2a7 Initializer < initialize
31f2a7 Server      A WSVR0023I: Server __adminServer open for e-business
```

The final trace, shown in Example 15-10, is typical of what we would expect to see if WebSphere, for some reason, fails to bind to the remote LDAP Directory Server. Note, however, that the initial search for the WebSphere Security Server ID identity user `wasadmin` appears to be successful. It is the subsequent `getRootDSE` sequence that fails, eventually throwing a `javax.naming.NameNotFoundException` with a failing LDAP error code 32 - No Such Object.

Example 15-10 When the Base Distinguished Name is not set

```
24c4a3 LdapRegistryI D Authenticating wasadmin
24c4a3 LdapRegistryI D Searching for users
24c4a3 LdapRegistryI > getUsers wasadmin
24c4a3 LdapRegistryI D filter = (&(uid=wasadmin)(objectclass=inetOrgPerson))
24c4a3 LdapRegistryI > getRootDSE
24c4a3 LdapRegistryI < getRootDSE
24c4a3 LdapRegistryI < getUsers
24c4a3 LdapRegistryI D Found user cn=wasadmin
24c4a3 LdapRegistryI D Authenticated with cn=wasadmin
24c4a3 LdapRegistryI > getGroupsForUser
24c4a3 LdapRegistryI D filter = (|(&(objectclass=groupofnames)(member=cn=wasadmin))(&(objectclass=groupofuniqueNames)(uniquemember=cn=wasadmin)))
24c4a3 LdapRegistryI > getRootDSE
24c4a3 LdapRegistryI < getRootDSE
24c4a3 LdapRegistryI > disconnect
24c4a3 LdapRegistryI < disconnect
24c4a3 LdapRegistryI > getRootDSE
24c4a3 LdapRegistryI < getRootDSE
24c4a3 LdapRegistryI > disconnect
24c4a3 LdapRegistryI < disconnect
24c4a3 LdapRegistryI > getRootDSE
24c4a3 LdapRegistryI < getRootDSE
24c4a3 LdapRegistryI > disconnect
24c4a3 LdapRegistryI < disconnect
24c4a3 LdapRegistryI < getGroupsForUser
      javax.naming.NameNotFoundException:
      [LDAP: error code 32 - No Such Object]
```

The problem indicates that the LDAP Base Distinguished Name was not set correctly under the WebSphere Global Security settings.

Note: In all the examples documented in this section, we omitted the respective timestamps.



IBM WebSphere Application Server and LDAP

The LTPA authentication mechanism in WebSphere can validate the user authentication against a third-party Lightweight Directory Access Protocol (LDAP) server.

IBM WebSphere Application Server 4.0 supports the following LDAP servers:

- ▶ IBM SecureWay Directory
- ▶ Netscape Directory Server
- ▶ Lotus Domino versions 4.6 and 5.0
- ▶ Microsoft Active Directory

This chapter discusses how to implement WebSphere security with the above-mentioned LDAP servers, giving an explanation of the necessary setup steps.

For more information about individual product installation, refer to the corresponding product installation guides.

16.1 SecureWay Directory Server

The use of IBM SecureWay Directory Server as the LDAP server with WebSphere can be an option for those companies that have different application environments and want to share a separate LDAP directory for authentication and authorization.

The IBM SecureWay Directory server provides a scalable, high performance directory service using IBM DB2 as the underlying database to provide pre-LDAP operation transaction integrity, high performance operations, and on-line backup and restore capability. For more details of the SecureWay Directory Server features refer to the IBM Redbook *LDAP Implementation Cookbook*, SG24-5110.

In our scenario, we demonstrate the use of the IBM SecureWay Directory Server as the LDAP server for supporting WebSphere authentication. The purpose of this section is to explain how to install and configure the SecureWay Directory and how to configure WebSphere to utilize it.

16.1.1 Installing and configuring the IBM SecureWay Directory

Installing the server

We used the following products' versions in the example documented over the next few pages:

- Windows 2000 with Service Pack 2.
- 1 GB RAM and 16 GB hard disk space.
- IBM SecureWay Directory Server V3.2.1 for Windows.
- IBM DB2 Universal Database V7.2.1, Enterprise Edition for Windows.
- IBM HTTP Server 1.3.19 for Windows as the Web server.
- IBM GSKit 4.01 package for SSL capability.
- Microsoft IE 5.5 for administering SecureWay.

Prior to installing the SecureWay Directory, the HTTP Server and the DB2 must be installed and running on the machine. For detailed instructions about how to install these products, refer to the corresponding installation documentation.

Note: The following are the *minimum* system and software requirements to install the SecureWay Directory server on Windows NT:

- ▶ Windows NT 4.0 with Service Pack 4 or later.
- ▶ A minimum of 64 MB RAM (128 MB is strongly recommended).
- ▶ One of the following Web servers (or later versions), installed and configured:
 - IBM HTTP Server 1.3.12
 - Lotus Domino Go Webserver(TM) 4.6.2.6
 - Lotus Domino Enterprise 5.0.2b Webserver(TM)
 - Microsoft Internet Information Server 4.0
 - Apache Server 1.3.12
 - Netscape FastTrack Server 3.01
 - Netscape Enterprise Server 3.6.3, 4.0

If a supported Web server is not detected on your system, the installation process automatically installs the IBM HTTP Server 1.3.12 that is included in the SecureWay Directory package.

- ▶ DB2(R) Universal Database for Windows NT, Personal, Workgroup, or Enterprise edition. The minimum supported level is DB2 Version 5.2 with FixPak WR09084. DB2 V6.1 is included with the IBM SecureWay Directory and is installed if a supported version of DB2 is not detected on your system.
- ▶ Microsoft Internet Explorer (MS IE) V4.0 plus service pack 1 or higher, or Netscape Navigator V4.07 or later (V4.08 is recommended), or Netscape Communicator V4.7, V4.8 or later for administering the SecureWay Directory.
- ▶ The Global Security Kit (Gskit) is an optional software package, included in the IBM SecureWay Directory software, that is required only if Secure Sockets Layer Security (SSL) is used. For SSL capability, the IBM Gskit V4.01 package needs to be installed on the system.
- ▶ Approximately 135 MB of disk space for both the SecureWay Directory and DB2.

The installation of the SecureWay Directory uses the Windows InstallShield that guides you through four general installation processes, which are:

- ▶ Language selection
- ▶ Destination location
- ▶ Components to configure
- ▶ Restarting the computer

To begin installing SecureWay Directory V3.2, perform the following steps:

1. Depending on whether you are installing locally from a CD or remotely from the network, select the CD-ROM drive and click the **setup.exe** icon.
2. When the Language panel is displayed, select the language you want to use and click **OK**.
3. Click **Accept** at the Software License Agreement.
4. Click **Next** in the Welcome LDAP Setup program window.
5. The *Installed applications* panel is displayed, telling you which products are installed already on the machine. In our case, the IBM HTTP Server and DB2 were pre-installed. Click **Next**.
6. Select the components to install:
 - **Express**: for installing all components that are not already installed.
 - **Custom**: for choosing the products individually to install.Select **Custom**.
7. Choose the target directory. By default, the product will be installed under the following directory: C:\Program Files\IBM\LDAP. Click **Next**.

Note: It is recommended that you use a shorter path and no spaces in directory names.

8. The Custom Installation window will be displayed as shown in Figure 16-1.

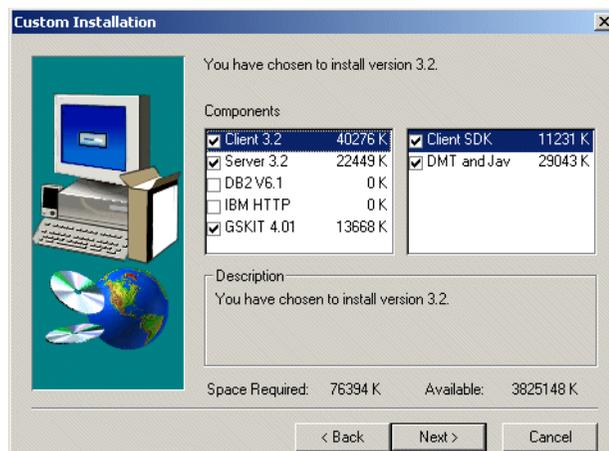


Figure 16-1 SecureWay Custom Installation panel

Select the components to install and then click **Next**.

9. Select the program folder where you want to add the program icons. By default, this is the **IBM SecureWay Directory**.
10. Select the components to configure as shown in Figure 16-3 on page 464:
 - Set the directory administrator name and password.
 - Create the directory database.
 - Configure a Web Server for directory administration.

Click **Next**. See the next section for more details about the configuration of the IBM SecureWay Directory.

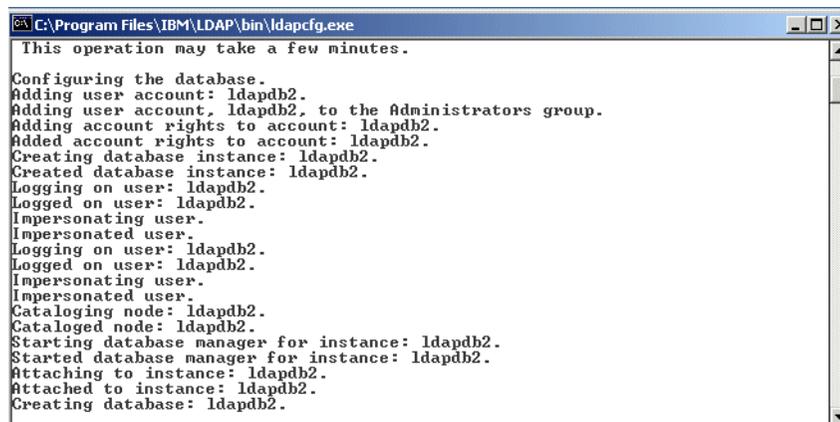
11. Once the configuration is finished, the *Start Copying Files for SecureWay Directory and Client SDK* panel displays the following information:

```
IBM SecureWay will be installed to the following directory: C:\Program
Files\IBM\LDAP
Target Folder: IBM SecureWay Directory
Password for administrator DN cn=root will be set.
LDAP UCS-2 (UTF-8) DB2 database will be created on drive C.
Web server IBM HTTP Server will be configured using file:C:\IBM HTTP
Server\conf\httpd.conf.
GSKIT 4.01 will be installed in C:\Program Files\IBM\GSK4.
```

Click **Next** to begin copying the files onto the machine.

12. When the setup is complete, a window is displayed prompting you to restart the computer.

If the configuration of the Directory server has been performed during the installation process, the LDAPDB2 database will be created automatically after you restart the machine, as shown in Figure 16-2 .



```
C:\Program Files\IBM\LDAP\bin\ldapcfg.exe
This operation may take a few minutes.
Configuring the database.
Adding user account: ldapdb2.
Adding user account, ldapdb2, to the Administrators group.
Adding account rights to account: ldapdb2.
Added account rights to account: ldapdb2.
Creating database instance: ldapdb2.
Created database instance: ldapdb2.
Logging on user: ldapdb2.
Logged on user: ldapdb2.
Impersonating user.
Impersonated user.
Logging on user: ldapdb2.
Logged on user: ldapdb2.
Impersonating user.
Impersonated user.
Cataloging node: ldapdb2.
Cataloged node: ldapdb2.
Starting database manager for instance: ldapdb2.
Started database manager for instance: ldapdb2.
Attaching to instance: ldapdb2.
Attached to instance: ldapdb2.
Creating database: ldapdb2.
```

Figure 16-2 Creating LDAPDB2 database during IBM SecureWay configuration

Configuring the server

The SecureWay Directory Server configuration can be performed during the installation process or by using the SecureWay Directory Server Administration tool **1dapxcfg** (graphical interface) or the **1dapcfg** command-line utility. The later two options can be undertaken after the installation is complete. The utilities can be used when the initial configuration needs to be changed or reset.

The configuration process consists of three parts:

1. Setting the directory administrator name and password.
2. Creating the directory database.
3. Configuring a Web Server for directory administration.

All of these steps can be performed at the same time by selecting all of the options. Alternatively, they can be done separately, but *all* must be completed prior to starting SecureWay for the first time.

To manually configure the IBM SecureWay Directory server, follow these steps:

1. Select **Start -> Programs -> IBM SecureWay Directory -> Directory Configuration** or type **1dapxcfg** at a command prompt (C:\Program Files\IBM\ldap\bin is included in the PATH of the machine during the installation process). The configuration utility window will be displayed as shown in Figure 16-3:

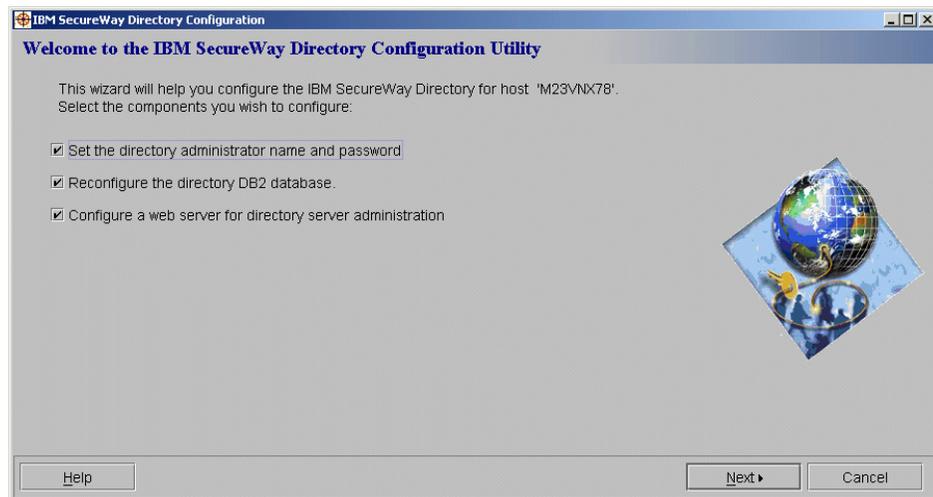


Figure 16-3 IBM Security Directory Configuration Utility

2. Select all the components and click **Next**.

3. Type the Administrator Distinguished Name (DN) or accept the default DN (cn=root) and set a password. Click **Next**.



Figure 16-4 Administrator DN and password for the IBM SecureWay Directory

4. Choose **Create a default LDAPDB2 database** and select **Create a native language DB2 database UTF-8** (UCS Transformation Format) to create the default DB2 database in the Universal Character Set for the directory server; then specify a location for this database.

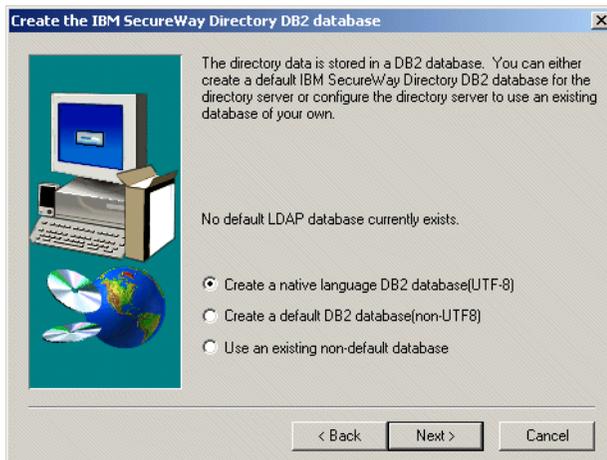


Figure 16-5 Creating the IBM SecureWay Directory DB2 database

This will create a new database instance named LDAPDB2 after the configuration completes as shown in Figure 16-2 on page 463.

5. Click **Next**.

Note: The directory configuration tool will also prompt you to create a database for change log support. It will create a new database called the change log (LDAPCLOG) for recording a log of changes made to the main directory. Select this option if you want to enable this feature.

Important: If you already have an LDAPDB2 database and want to use the directory configuration tool to create a new database, all data will be lost. It is recommended that you save the data by using the **db2idif** command. For additional information, refer to the Server Administration online help under Command Line Utilities.

It is also possible to configure the database using the SecureWay Web-based Administration interface by selecting **Database -> Configure** in the left pane and with the Directory Server stopped.

6. Select the IBM HTTP Server from the list of Web servers and enter the full path name of the Web server configuration file.

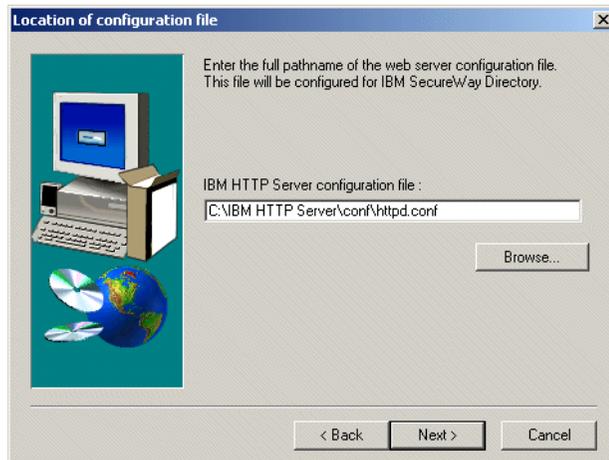


Figure 16-6 IBM HTTP Server configuration file location

Attention: Do not forget to restart the IBM HTTP Server after the SecureWay Configuration completes.

7. Click **Next** and the configuration summary window will be displayed. Click **Configure** to begin the configuration.

8. Restart the machine and start the IBM SecureWay Directory. Before you start the SecureWay Directory using the Windows services, check that the following services are started:

- IBM HTTP Server
- DB2-LDAPDB2

Then start the IBM SecureWay Directory service.

It is possible also to start the SecureWay Directory using the Web-based interface by opening a Web browser and accessing the URL for Server Administration using the format: `http://<your server>/ldap`. Enter the Admin ID and password as shown on Figure 16-7.



Figure 16-7 Logging on to IBM SecureWay Directory Server Administration

Then select **Current State -> Start/Stop**.

There is an alternative method for configuring the IBM SecureWay Directory Server; this is using the `ldapcfg` command line tool. For more details about this method, read the *Installation and Configuration Guide* or run the command without specifying parameters to get the command syntax help.

16.1.2 Populating data entries in the IBM SecureWay Directory

Once the installation and configuration are finished, proceed to add new data entries into the directory. The steps to populate data entries in the directory are the following:

- ▶ Define a suffix
- ▶ Add user entries
- ▶ Add group entries

Defining a suffix

Before you can add entries to the database, it is necessary to define a suffix for that directory. A suffix is the starting point in the directory and specifies the Distinguished Name (DN) for the root of that tree. The LDAP server must have at least one suffix defined and can have multiple suffixes. Each entry added to the directory contains in their fully Distinguished Name (DN) a suffix that matches one of the server's suffixes defined on the server.

To define a valid suffix, it is possible to use the X.500 naming structure that will set the root of the directory to a specific organization in a specific country or to a specific organization and organizational unit:

o=ibm,c=us

where o represents Organization and c represents Country, and

ou=raleigh,o=ibm

where ou represents Organizational Unit and o represents Organization.

It is also possible to use the DNS naming model by using the *domainComponent* attribute:

dc=ibm.com

where dc represents a domain component, for example:

dc=itso,dc=ral,dc=ibm,dc=com

To add a suffix in the directory, follow these steps:

1. Open a Web browser, then access the URL for the Server Administration:
`http://<your server>/ldap.`
2. Once logged in as an administrator, click the **Add Suffixes** link at the top of the page or expand the **Settings -> Suffixes** folder then click **Add Suffixes**.

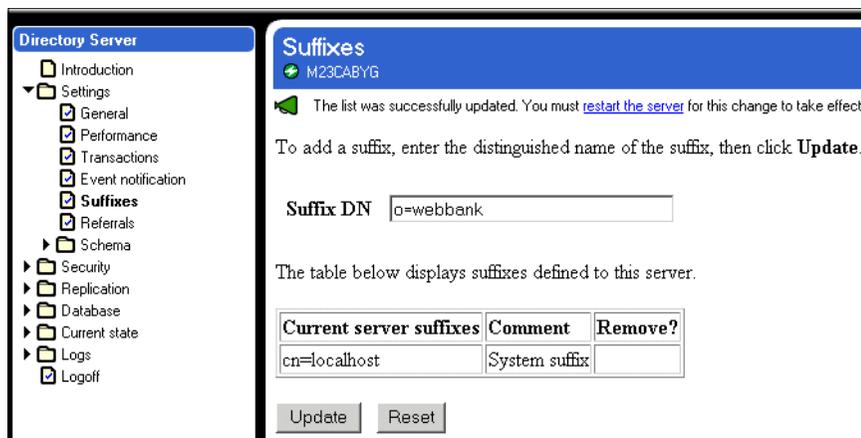


Figure 16-8 Adding a new suffix in the IBM SecureWay Directory Server Administration

For our example, using the X.500 methodology, we set the following suffix: o=webbank, where o represents an organization. Click **Update**, and the Suffix table list will be updated with the new suffix. At this point, it contains no data.

- Restart the server by clicking the **Restart Server** link on the top of the page so that the changes may take effect.

It is also possible to remove suffixes from the directory by clicking the box next to the suffix you want to delete. Removing a suffix eliminates access to all directory data beneath that suffix, but the data is not removed from the directory.

Current server suffixes	Comment	Remove?
cn=localhost	System suffix	
o=webbank		<input checked="" type="checkbox"/>

Figure 16-9 Removing a suffix in the IBM SecureWay Directory Server Administration.

To check the actual status of the server, expand **Current State -> Server Status**.

Adding user entries

To add new user entries in the directory, use the Directory Management Tool (DMT) or provide the data in an LDAP Data Interchange Format (LDIF) file.

The DMT is a Java-based graphical tool for browsing, checking and managing directory entries.

To add a new entry, follow these steps:

- Open the Directory Management Tool by clicking **Start -> Programs -> IBM SecureWay Directory->Directory Management Tool**.
- Click the **Add Server** button. The Add Server fields are displayed. Type in the Server Name and Port and then select **Simple** in the Authentication Type.
- Log on as the **cn=root** user and click **OK**.
- A Warning message box will be displayed, notifying you that the suffix created previously (o=Webbank) does not contain any data. Click **OK**.

Note: If an authenticated user is not introduced during the Add Server process, you will only have anonymous privileges; this means that it is only possible for you to browse the directory, but not to modify it. To authenticate yourself, select **Server->Rebind** from the menu in the left-hand pane.

The directory tree is now open with no suffix appearing in the tree, as shown in Figure 16-10.

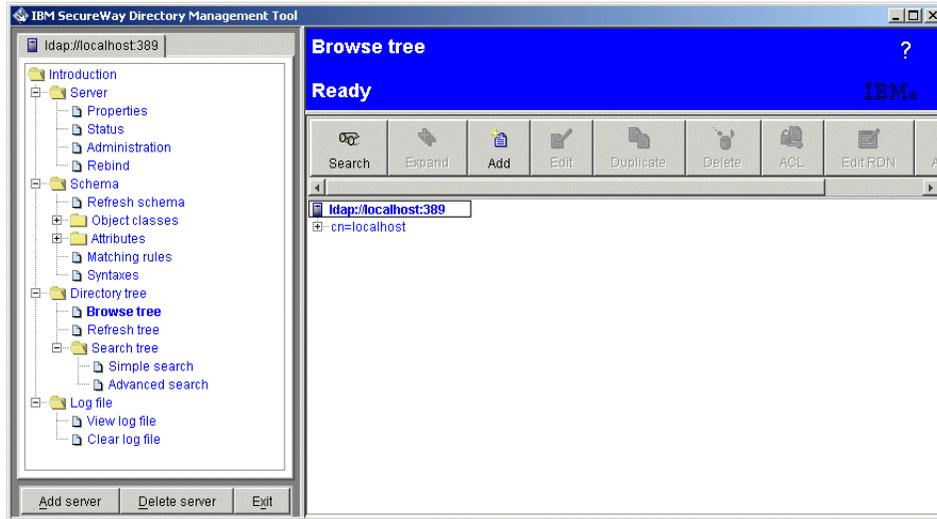


Figure 16-10 Directory tree browser open

- To add a new entry in the directory, click the **Add** button in the toolbar. Select **Organization** as the Entry Type and **o=Webbank** as the Entry RDN (Relative Distinguished Name), as shown in Figure 16-11.

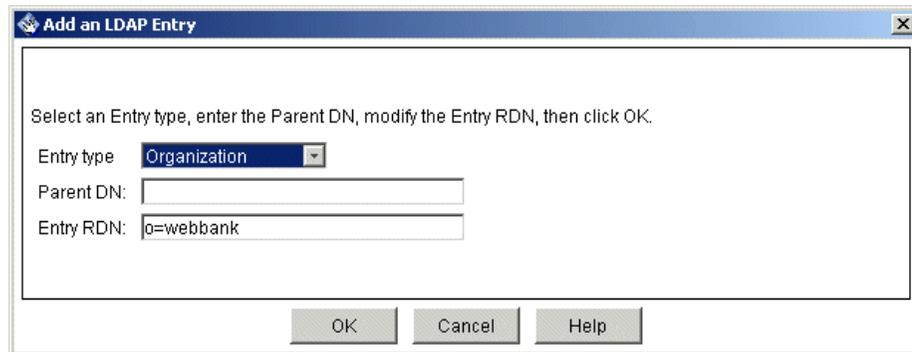


Figure 16-11 Adding an Organization RDN

- Click **OK** and a new window appears for setting attributes to the new RDN entry. All the fields marked in bold style are mandatory.

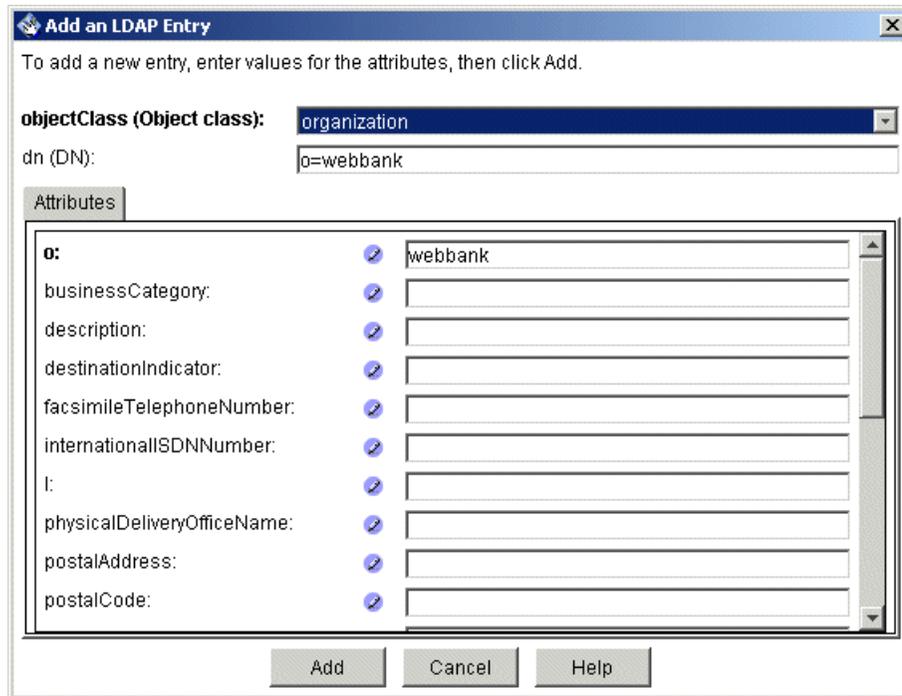


Figure 16-12 Setting new attributes to the organization RDN

7. Click **Add**. The new organization entry should appear in the directory tree after clicking **Directory** -> **Refresh Tree** as shown in Figure 16-13.

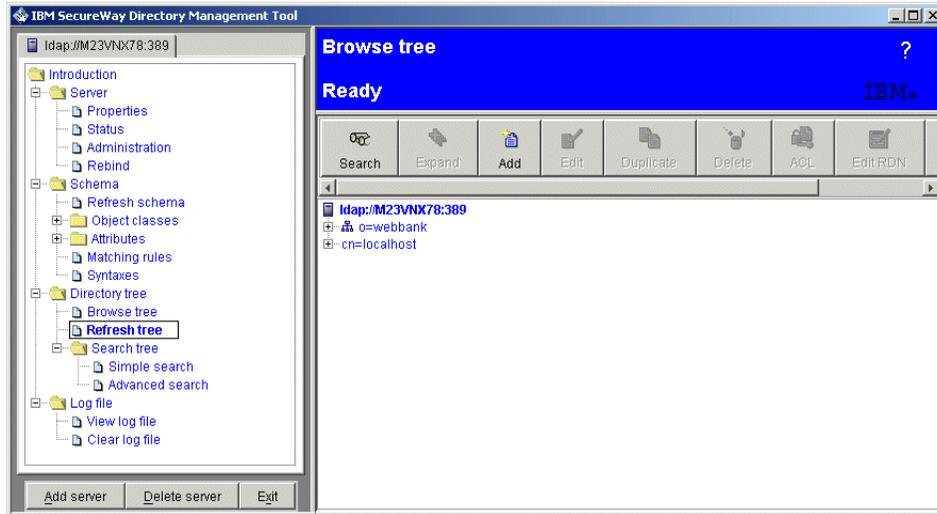


Figure 16-13 New organization in the Directory tree after refreshing

8. Next, we create a new Country entry to include the new users beneath it. Select the organization **o=webbank** and click the **Add** button in the toolbar. Select **Country** as the Entry Type and **c=US** as the Entry RDN (Relative Distinguished Name) as shown in Figure 16-14.

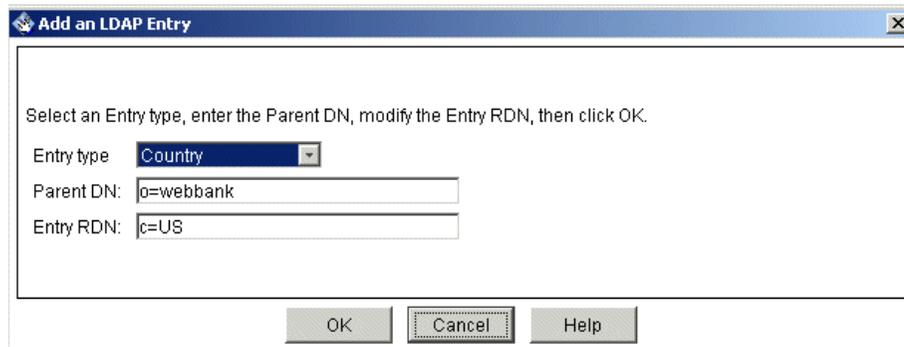


Figure 16-14 Adding a new Country entry

9. Click **OK**; a new window appears for setting attributes to the new RDN entry.

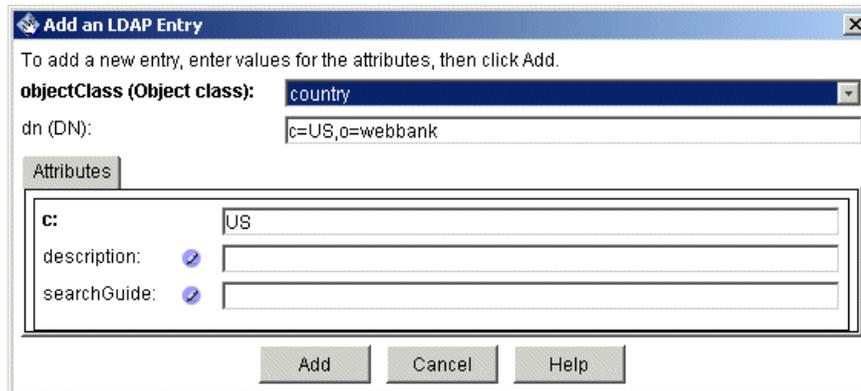


Figure 16-15 Setting new attributes to the Country RDN

10. Click **Add**. The new country entry should appear in the directory tree beneath the organization entry (o=webbank) after clicking **Directory -> Refresh Tree**.
11. Now, proceed to add new users to the new organization created previously. We are going to add a new administrative user in order to set a Security Server ID in the WebSphere Global Security settings, for use as the user ID under which the server runs for security purposes.

Select the country **c=US** and click the **Add** button on the tool bar. Select **User** as the Entry Type and type **cn=wasadmin** as the Entry RDN.

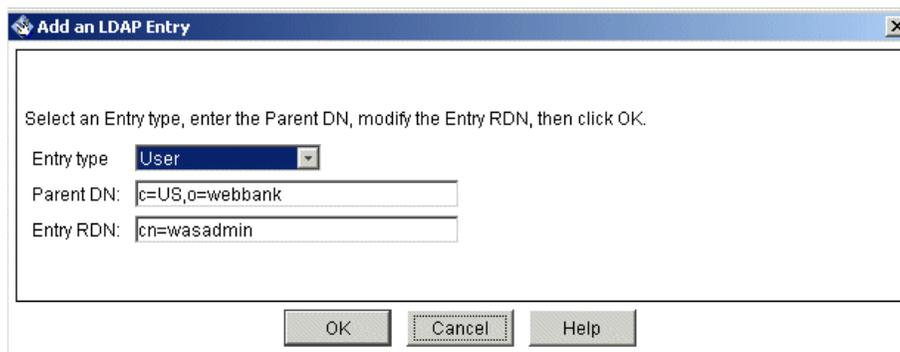


Figure 16-16 Adding a new user

12. Click **OK**; a new window appears for setting attributes to the new RDN entry, as shown in Figure 16-17.

The Distinguished Name (DN) is the fully qualified name for the user. The default DN is the Parent DN plus the Entry RDN. Enter the Last Name (required for adding new users) and a password. Include any other information in the Business and Personal tabs.

13. Switch to the **Other** tab and supply the **UID** to allow the user to authenticate himself with this value instead of using the full DN.

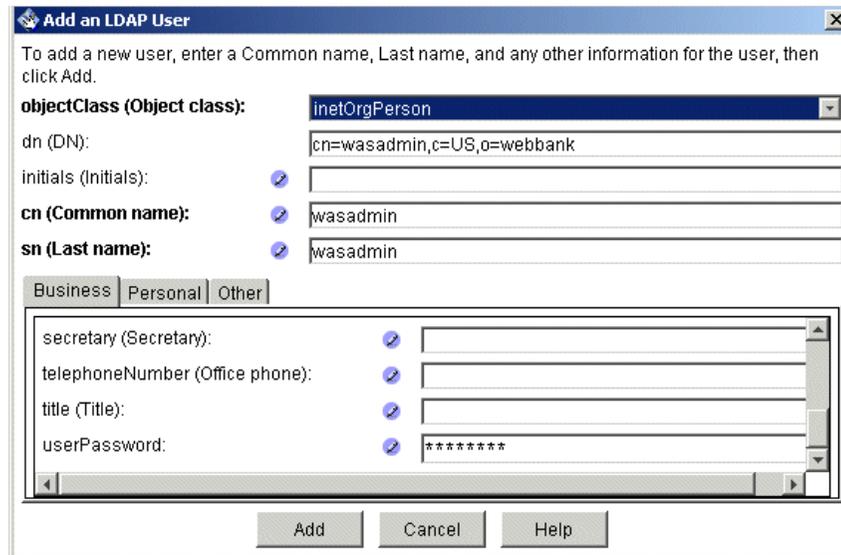


Figure 16-17 Setting attributes for the new user

14. Click **Add**. The new user entry should appear in the directory tree beneath the organization entry (c=us,o=webbank) after clicking **Directory->Refresh Tree**, as shown in Figure 16-18.



Figure 16-18 Directory tree with the new user

15. Create new application users following the procedure explained above, repeating steps 9 to 11.

Adding group entries

To add new groups in the directory, it is possible to use the DMT (Directory Management Tool) or provide the data in an LDIF (LDAP Data Interchange Format) file. To add new groups, follow these next steps:

1. Select the country **c=US** and click the **Add** button in the toolbar.
2. Select **Group** as the Entry Type and type **cn=Employees** as the Entry RDN as seen in Figure 16-19; then click **OK**.

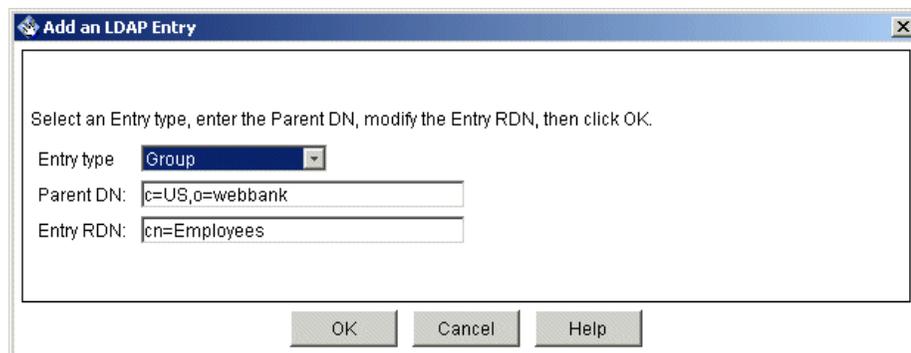


Figure 16-19 Adding a new group

3. The next window is used for setting attributes for the new RDN entry, as shown in Figure 16-20.

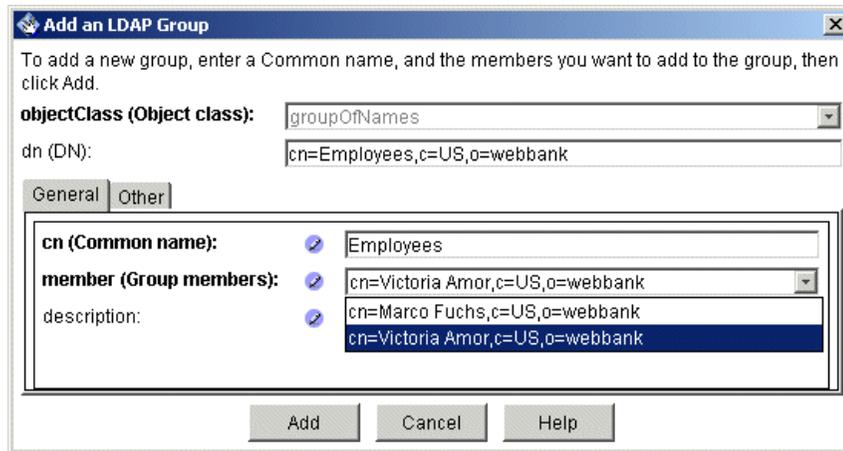


Figure 16-20 Setting groups attributes

The *cn:* and *member (Group member):* are mandatory fields; we assigned the following group members:

- cn=Victoria Amor, c=US, o=webbank
- cn=Marco Fuchs, c=US, o=webbank

For our example, we also created two new groups, repeating the steps above with the following values.

Table 16-1 New groups' values

Full DN	Common Name	Group Members
cn=Managers,c=US,o=Webbank	Managers	cn=James Roca,c=US,o=Webbank cn=Joanna Hodgson,c=US,o=Webbank
cn=Customers,c=US,o=Webbank	Customers	cn=Peter Kovari,c=US,o=Webbank

4. Click **Add**. The new group entries should appear in the directory tree beneath the organization entry (c=us, o=Webbank) after clicking **Directory** -> **Refresh Tree**, as shown in Figure 16-21.

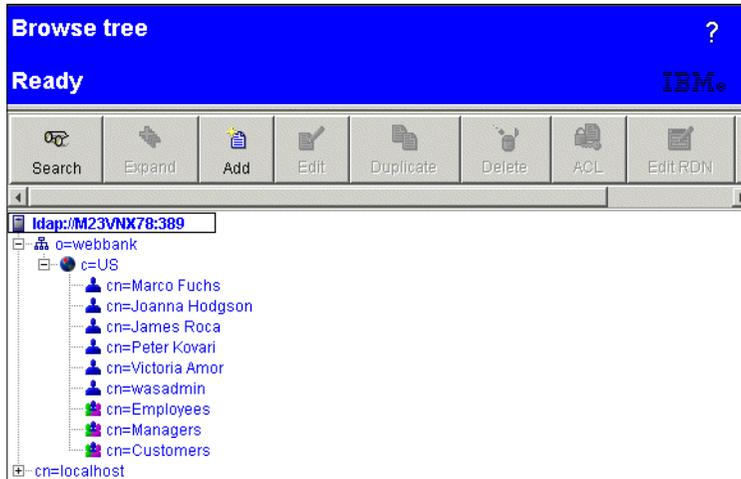


Figure 16-21 Directory tree with the new groups

As mentioned before, it is possible to add users and groups using an LDIF file, a standard format for representing LDAP entries in text form. This will be useful when the number of entries to add is very large. An example of an LDIF file is shown below.

Example 16-1 Sample LDIF file

```
dn: o=ibm, c=us
objectclass: organization
objectclass: top
o: ibm, c=us
o: ibm

dn: cn=Robert Red,o=ibm,c=us
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: ePerson
objectclass: top
objectclass: person
uid: red
userpassword: {iMASK}>15TEoHuM0v8qy9gcuXE+/AqnogmwX5NkRE7cGeHEG99ttyaTeIqiEX14
D7oB1wDJrDNBf1WveOF+DDHmyGnxFTIht+0trS9mAzHL0517tsdE3wL0aTbHJXYV8sc17GZD6Ghp
d/d1JzaSI2LD6+dZW/Tc1HLMUuM6/L<
sn: Red
cn: Robert Red
```

```
dn: cn=Manager,o=ibm,c=us
objectclass: top
objectclass: groupOfNames
cn: Manager
member: cn=Robert Red,o=ibm,c=us
```

To import the file, perform the following steps:

1. Open the browser and access the URL for Server Administration using the format `http://<your server>/ldap`. Once logged as a *root*, check that the Directory server is running.
2. Select **Database -> Import LDIF**.
3. Enter the name of the LDIF file on the LDAP server from which you want to import directory data, then click **Import** as shown in Figure 16-22.

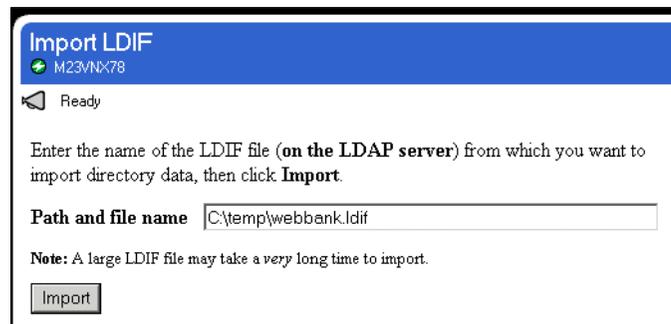


Figure 16-22 IBM SecureWay Administration GUI - Import Webbank.ldif file

4. Wait until a message appears indicating that the entries have been successfully added.
5. Use the DMT tool to verify the new entries.

16.1.3 Configuring WebSphere to use the SecureWay Directory Server

To configure WebSphere to use IBM SecureWay Directory as its User Registry, follow these next steps:

1. Start WebSphere Administrator's Console (WebSphere Admin Server V4.0 must be running).
2. Select **Console -> Security Center...** This will display the security settings for WebSphere. Select **Enable security** in the General tab.

3. Switch to the **Authentication** tab and specify the following settings:
 - Lightweight Third Part Authentication (LTPA).
 - Select the **LDAP** radio button. This will display the LDAP settings. Set the fields as follows:
 - **Security Server ID:** this field should contain the full DN or the value specified in the **UID** field included in the LDAP User Properties, under the **Other tab** information created in the previous steps for the WebSphere administrator user. This is the user ID that you will have to enter in order to start the WebSphere console once security is enabled.
 - **Security Server Password:** enter the user password set for the wasadmin user in LDAP.
 - **Host:** introduce the Host ID (IP address or DNS name) of the LDAP server.
 - **Directory Type:** select **SecureWay** as the LDAP server type.
 - **Port:** the port number, by default, is 389 if nothing is specified. If the Default SecureWay port number changes, it will be necessary to specify it in this field.
 - **Base Distinguished Name:** this field is required for the SecureWay directory. A Base DN indicates the starting point for LDAP searches of the directory service.
 - **Bind Distinguished Name:** the fully distinguished name used by WebSphere to bind with the LDAP server. If no name is specified, the administration server will bind anonymously.
 - **Bind password:** if a user is specified in the Bind Distinguished name, include the corresponding password here.
4. All the settings are illustrated in Figure 16-23 on page 480.

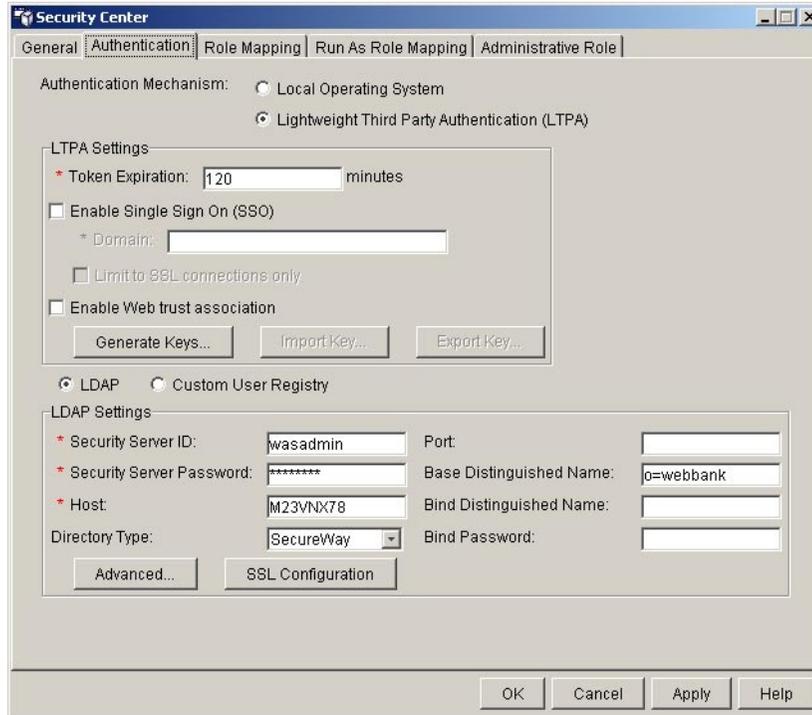


Figure 16-23 LDAP settings for IBM SecureWay in WebSphere security

Clicking the **Advanced** button lets you view the default LDAP advanced properties governing how WebSphere performs the query for the SecureWay Directory, as shown in Figure 16-24.

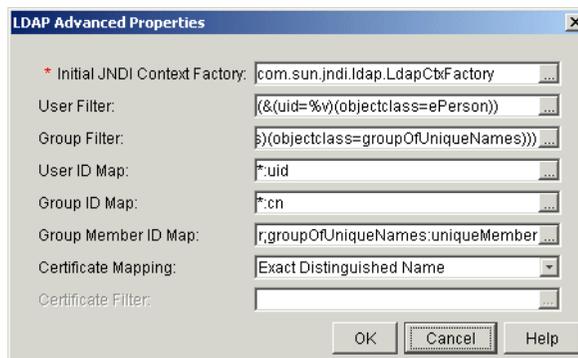


Figure 16-24 LDAP Advanced Properties window

The default SecureWay filter in LDAP Advanced Properties executes the following searches:

The User Filter finds a user entry in the SecureWay Directory with the attribute *uid*, belonging to the *ePerson* object class.

The Group Filter finds the groups' entries in the Secureway Directory with the attribute *cn*, belonging to the *groupOfNames* or *groupOfUniqueNames* object classes, set with the group name you are looking for. It is typically used for the assignment of a Security Role to a Group.

Note: *ePerson*, *groupOfNames* and *groupOfUniqueNames* object classes are part of the SecureWay LDAP schema categories; to find out more about the Secureway LDAP schema, refer to *LDAP Implementation Cookbook*, SG24-5110.

The User ID map and Group ID map are the filters that map the *short name* of a user or group to an LDAP entry; this information represents users or groups when they are displayed. LDAP returns the field specified by the User ID map and Group ID map (*uid* in the case of a user and *Common Name* in the case of groups) when the search performed by WebSphere is successful. WebSphere uses these returned values for comparison with permission lists.

The Group Member ID map identifies memberships of Users to Groups.

These default settings will be suitable for searching for Users and Groups in SecureWay Directory. When the LDAP advanced properties are modified, the Directory Type changes to **Custom** in the Security Center panel; this reflects the fact that the default settings for searching are no longer used.

5. Click **OK**. The Security Server ID and Security Server Password will be verified against the LDAP server.

Make sure that the IBM SecureWay Directory is running by checking the list of Windows' services on the directory server and pinging the IP address of the SecureWay Directory machine from the WebSphere machine.

The first time the security is enabled, it will prompt for an LTPA password. The LTPA password is used to protect the set of encryption keys generated for use in a Single Sign-On environment. For more information about Single Sign-On, refer to the Chapter 14, "Single Sign-On" on page 393.

When the process is complete, a warning message appears, stating that changes will not take effect until the admin server is restarted.

6. Restart the Administration Server by selecting the node, then right-clicking it and selecting **Restart** in the resulting context menu.

7. When the server is open for e-business, start the Administrator's Console. You will be prompted for the administrator user ID (Security Server ID) and password (Security Server Password).

The window is shown in Figure 16-25.



Figure 16-25 WebSphere Administrator's Console Password prompt

16.2 Lotus Domino 5.0

Domino, as a common user registry for WebSphere, can be an option for those organizations that already have Domino as a main application server and want to exploit the directory architecture already in place, using WebSphere as a complement.

You can set up a Domino server to run the Lightweight Directory Access Protocol (LDAP) service to enable LDAP clients to search for and modify information in the Domino Directory. For more details about Domino LDAP services features, refer to the *Domino R5 Administration Help and Administrator's Guide*.

The procedure to use Domino with WebSphere as an LDAP server involves the following steps:

- ▶ Configure Domino Server to run the LDAP service.
- ▶ Configure WebSphere to use the Domino Directory as its user registry.

16.2.1 Configuring the Domino Server to run the LDAP service

Make sure that you have installed and configured a Domino Server. For detailed instructions, refer to the installation manuals.

The only necessary steps to set up the Domino Server for LDAP service are as follows:

- ▶ Configure the LDAP port in the server configuration document.
- ▶ Start the Domino LDAP server task.

Configuring the LDAP port in the server configuration document

To configure the LDAP port in the server configuration document, perform the following steps:

1. Start the Domino Administration client logged in as a user with administrator privileges, then open the Domino Server from the left server bookmark pane. Click the **Configuration** tab.
2. Select **Server -> Current Server Document**.
3. Click **Ports -> Internet Ports** and go to the **Directory** tab to display the LDAP port settings. The default LDAP TCP/IP port and SSL port are shown:
 - LDAP: 389
 - LDAPS: 636

In our present scenario and for testing purposes, we are going to use the default settings.

4. Make sure that at least the TCP/IP port is enabled.
5. Decide which type of authentication options are better suited to your environment, choosing **Yes** or **No** to allow a Name and Password for authentication of clients connecting over TCP/IP, and **Yes** or **No** if you want to allow anonymous connections over TCP/IP.

In a secure environment, usually the TCP/IP Port is disabled and the SSL port is enabled with at least one of the Authentication options selected as **Yes**:

- *Client certificates*: allow the use of X.509 client certificates for authentication
- *Name and Password*: allow the name and password for authentication.
- *Anonymous*: allow anonymous connections over SSL.

Note: If you allow anonymous connections, you can configure which fields anonymous LDAP clients can search in Domino by editing the Directory Settings Document. Be aware that the Anonymous entry in the Domino Directory ACL does not control anonymous LDAP access. Consult the *Domino R5 Administration Help and Administrator's Guide* for further details.

For more details on how to implement SSL in Domino, refer to Section 11.6.13, “SSL and Lotus Domino LDAP” on page 317.

Every time you modify the LDAP parameters in the Server Document, it is necessary to restart the LDAP task on the Server.

To restart the LDAP task from the Domino Administration Client, select the **LDAP task** from the task pane in the **Status** tab (**Server-> Status**), right-click the highlighted task and select **Stop**.

Start the LDAP task again in the server following the procedures described next.

To restart the LDAP task from the Domino console, first enter the **tell ldap quit** and then the **load ldap** commands.

Starting the Domino LDAP server task

There are three ways to start the Domino LDAP server task:

- ▶ Using the Domino Console by entering the **load ldap** command.
- ▶ Adding the *ldap* name in the `Server Task=` line, in the `notes.ini` file, to begin the task every time the server is started.
- ▶ Using the Domino 5 Administration Client:
 - a. Start the Administration Client and open the Domino Server from the left server bookmark pane, then click **Status** tab (**Server->Status**).
 - b. Open the task toolbar located on the right side of the task pane and select the **Start...** action. Choose the LDAP Server entry and click **Start Task**.
 - c. Click the **Done** button to exit the tasks dialog.

Note: You must have administration privileges in order to start the LDAP server task.

16.2.2 Configuring WebSphere to use the Domino Directory

In order to set up WebSphere security settings using the Domino Directory as the LDAP server, it is necessary to specify a Security Server ID, that is, the user ID under which the server runs for security purposes. This user ID should be any user registered in the Domino Directory (Notes or Internet/intranet users) with an Internet password set. We recommend that you create a new specific user in the registry to be used by WebSphere.

For that purpose, start the Domino R5 Administration with a notes ID having at least author access for creating new documents in the Domino Directory; make certain the *UserCreator* role is selected and follow these instructions:

1. Open the Domino Server from the left server bookmark pane and select the **People and Groups** tab.
2. Click the **Add Person** button.
3. Add a new user as shown in Figure 16-26 with the Short Name/UserID and Internet password set.

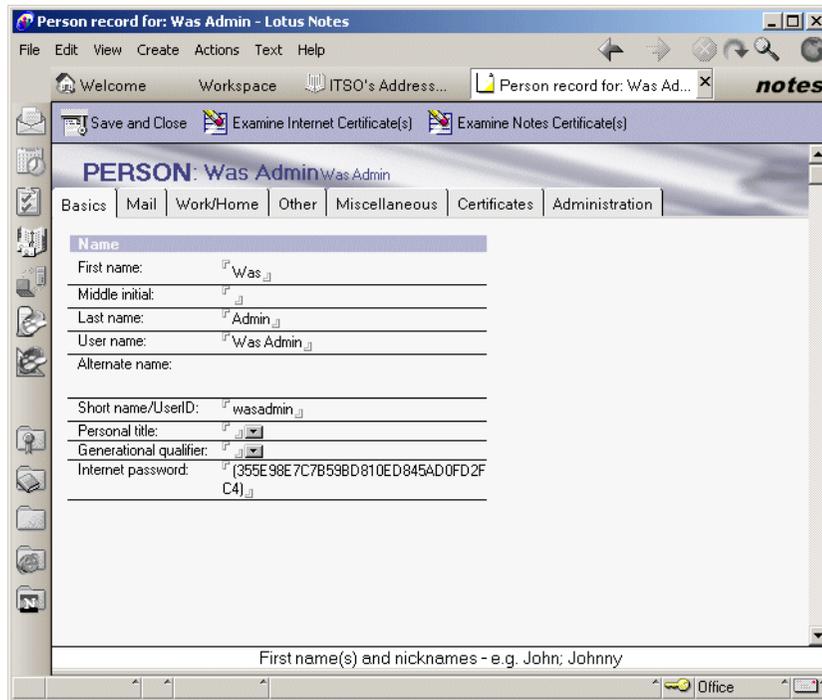


Figure 16-26 WebSphere Administration Person Document in Domino Directory

4. Save and close the document.

Once the user is created, ensure that it is possible to look for it in Domino using the LDAP protocol. Domino provides a command-line search utility that allows you to use LDAP to search entries in the Domino Directory on a server that runs the LDAP service, or search entries in a third-party LDAP directory.

This tool is included in the Domino server and Notes client software.

Note: To use the *ldapsearch* tool for searching against a Domino Directory, the LDAP task in the Domino Server must be started and the notes.ini file must be included in the machine system's Path environment variable where ldapsearch will be executed.

In a command line window from the Domino server or Notes client, perform the search by entering the **ldapsearch** command as shown in Figure 16-27.

```
C:\WINNT\System32\cmd.exe

C:\>\cd lotus
C:\Lotus>\cd notes
C:\Lotus\Notes>ldapsearch -v -h M23X2640 "uid=wasadmin"
ldap_open( M23X2640, 389 )
filter_pattern: uid=wasadmin
returning: ALL

*** Filter is: <uid=wasadmin> ***
CN=Was Admin
cn=Was Admin
shortname=wasadmin
uid=wasadmin
objectclass=top
objectclass=person
objectclass=organizationalPerson
objectclass=inetOrgPerson
objectclass=dominoPerson
givenname=Was
sn=Admin
1 matches

C:\Lotus\Notes>
```

Figure 16-27 LDAP search results for the wasadmin user

For more details about parameter attributes and search filters for use with the ldapsearch utility, refer to the *Domino R5 Administration Help and Administrator's Guide*.

To configure WebSphere to use Domino as its User Registry, follow these steps:

1. Start the WebSphere Administrator's Console (WebSphere Admin Server V4.0 must be running).
2. Select **Console -> Security Center...** This will display the global security settings for WebSphere. Select **Enable security** under the **General** tab.
3. Click the **Authentication** tab and specify the following settings:
 - Lightweight Third Part Authentication (LTPA)
 - Select the **LDAP** radio button. This will display the LDAP settings. Set the fields as follows:
 - **Security Server ID:** this field must contain the value specified in the Short Name/User ID field in the Person Document of the Domino Directory created in the steps above for the WebSphere administrator. This is the user ID that will have to be used for login to start the WebSphere Administrator's Console once security is enabled.
 - **Security Server Password:** enter the Internet password set for the wasadmin user in this document.
 - **Host:** the Host ID (IP address or DNS name) for the Domino server.
 - **Directory Type:** Select **Domino R5.0** as the LDAP server.

Leave all other fields blank, taking into account the following considerations:

- **Port:** the port number, by default, is 389 if none is specified. If the Default Domino LDAP port number changes, it will be necessary to specify it in this field.
- **Base Distinguished Name:** this field is required for all the LDAP directories except for Domino. If a Base DN is specified, it will *not* be able to grant permissions to individual Web users for resources managed by the IBM WebSphere Application Server.
- **Bind Distinguished Name:** the distinguished name used when WebSphere binds with the Domino server. If no name is specified, the administration server will bind anonymously.
- **Bind password:** if a user is specified in the Bind Distinguished name, include the corresponding password here.

All the settings are illustrated in Figure 16-28.

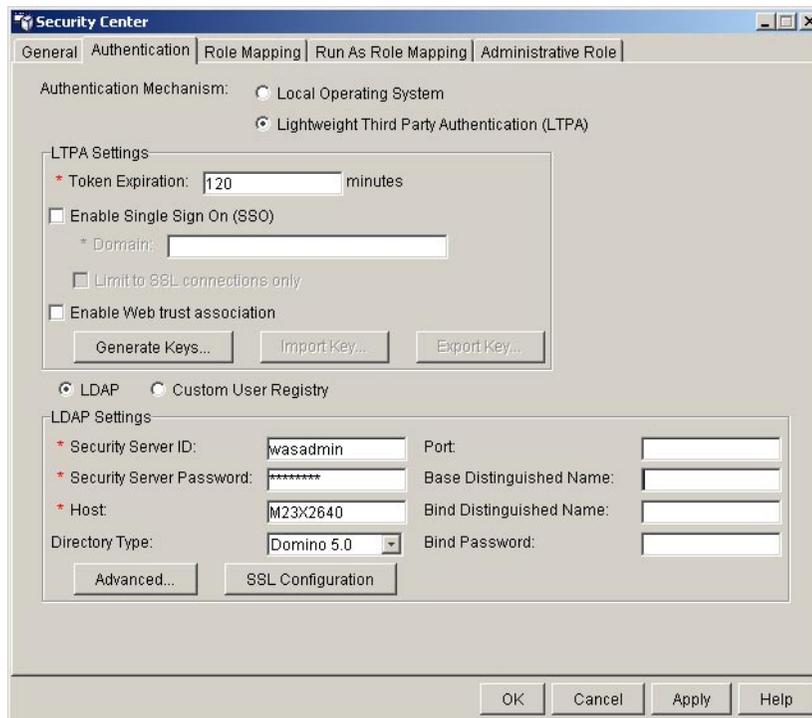


Figure 16-28 LDAP settings for Domino In WebSphere Security

Click the **Advanced** button; this brings up the default LDAP Advanced Properties window shown in Figure 16-29; you can now see how WebSphere searches the Domino Directory.

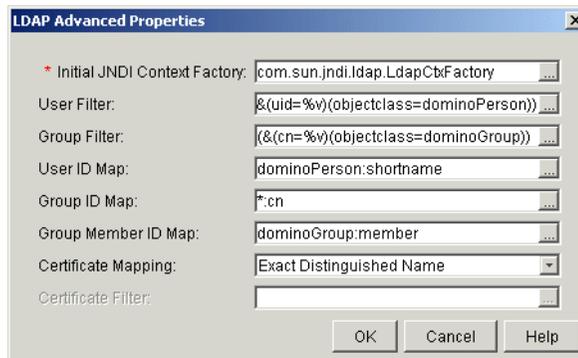


Figure 16-29 Domino LDAP Advanced Properties window

The default Domino filters in LDAP advanced properties execute the following searches:

The User Filter find a user entry in the Domino Directory with the attribute *uid*, belonging to the *dominoPerson* object class, set with the name entered at authentication. It is typically used for the assignment of a Security Role to a User.

The Group Filter find the groups entries in the Domino Directory with the attribute *cn*, belonging to the *dominoGroup* object class, set with the group name you are looking for. It is typically used for the assignment of a Security Role to a Group.

Note: *dominoPerson* and *dominoGroup* object classes are part of the Domino LDAP schema; to find out more about the Domino LDAP schema, use the **ldapsearch** utility or refer to the Domino LDAP Schema Database.

The User ID map and Group ID map are the filters that map the short name of a User or Group to an LDAP entry; this information represents Users or Groups when they are displayed. LDAP returns the field specified by the *User ID map* and *Group ID map* (short name in case of a user and common name in case of groups) when the search performed by WebSphere has been successful. WebSphere uses these returned values for comparison with permission lists.

The Group Member ID map identifies membership of Users to Groups.

In most cases, these default settings are suitable for searching Users and Groups in a Domino Directory. Notice that modifying the LDAP advanced properties changes the Directory type to **Custom** to reflect the fact that the entries are no longer the default for the Domino directory.

4. Click **OK** in the Security Center window. The user ID and the password will be verified against the information within the Domino Server.

Make sure that the Domino server is running and that the LDAP task is started by entering the **show task** command in the Domino Console. This will display the status of the active server tasks, as shown in Figure 16-30.

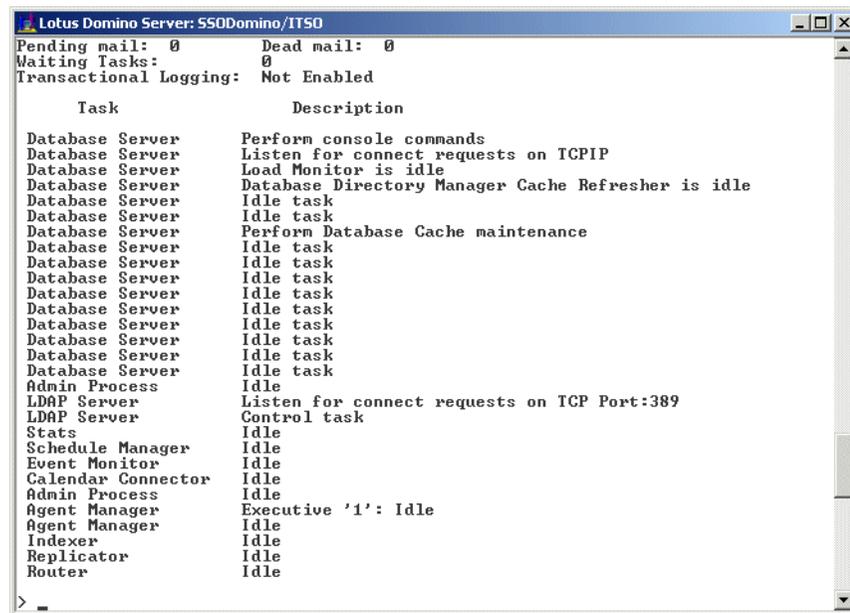


Figure 16-30 Show Task Command response in the Domino Server Console

The first time that security is enabled, it will prompt for an LTPA password.

The LTPA password is used to protect the set of encryption keys generated for use in a Single Sign-On environment.

For more information about Single Sign-On, refer to Chapter 14, “Single Sign-On” on page 393.

When the process is complete, a warning message will be displayed, stating that changes will not take effect until the admin server is restarted.

5. Restart the Administration Server by selecting the node, then right-clicking it and selecting **Restart** in the resulting context menu.

6. When the server is open for e-business, start the Administrator's Console. You will be prompted for the administrator user ID (Security Server ID) and password (Security Server Password).

16.3 Netscape Directory Server

This section explains how to configure WebSphere to use the Netscape Directory Server 4.2 as an LDAP server. To do this, follow these steps:

- ▶ Add a new administrative user to the Netscape Directory for use with WebSphere.
- ▶ Configure WebSphere to Use the Netscape Directory as its user registry.

16.3.1 Adding a new user

The following steps will show how to add a new user to the directory.

1. Make sure that the Netscape Directory Server 4.2 for Windows NT is installed and configured on your system. For detailed instructions, refer to the *Product Installation Guide*.
2. Start the Netscape Console to connect to the Administration Server in your network.

From the Start menu, choose **Programs**. Then, from the Netscape Server Family Program Group, choose **Netscape Console 4.2**.

3. When the Netscape Console login window appears, type the user name and password for the administrator user, and then specify the URL for the Administration Server you want to access.



Figure 16-31 Netscape Console login window

The administration user is defined during the installation process, as is the administration port. Click **OK**.

4. The Netscape console is now displayed (see Figure 16-32).

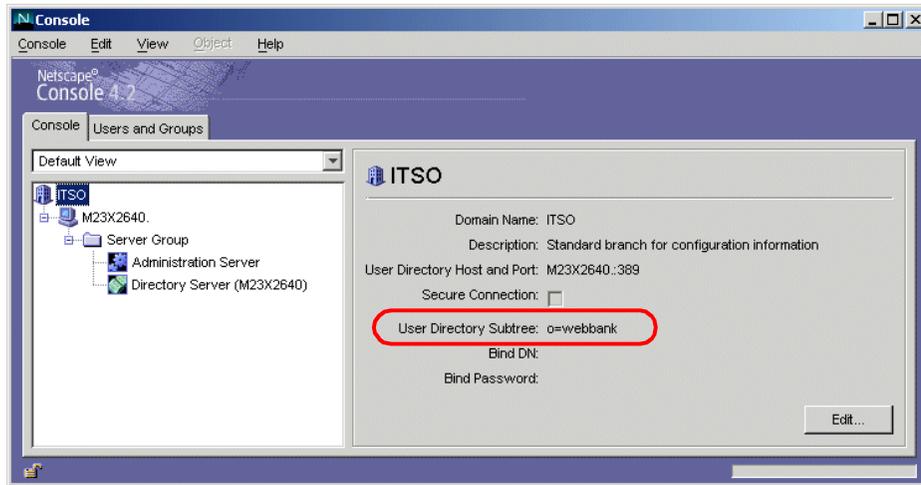


Figure 16-32 Netscape Console window

The **Console** tab shows all the information included in the Netscape Topolog, such as information about the Administration Domain (ITSO), the Server Group or the individual server (Administration server or Directory server). Notice that the User Directory Subtree (o=Webbank) will be used as the base DN in the WebSphere Global Security Settings.

5. Switch to the **Users and Groups** tab; you will see the window used for adding or modifying a user, a group, or an organizational unit (see Figure 16-33).

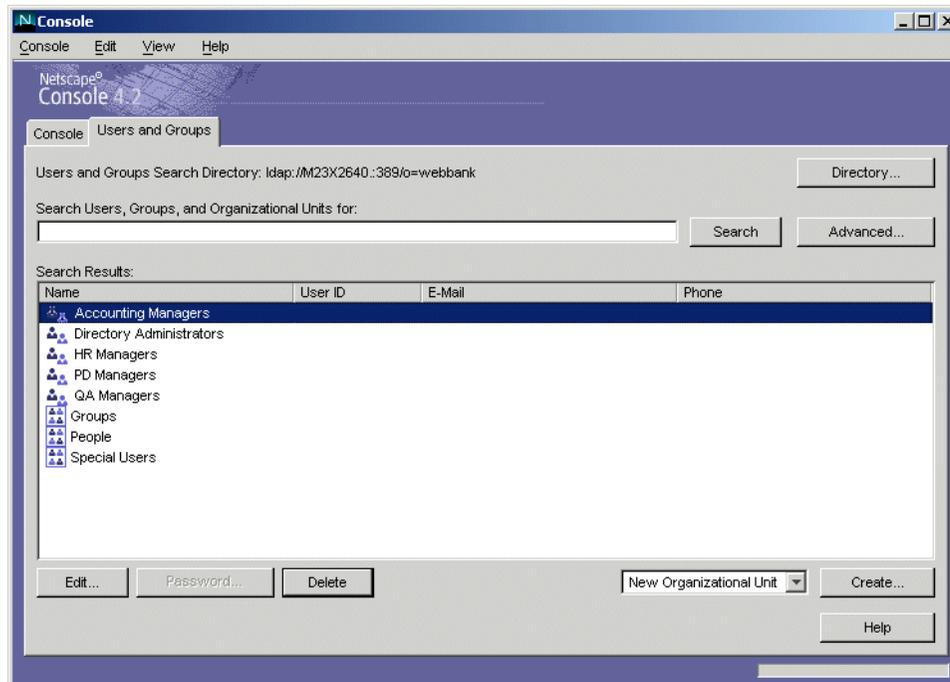


Figure 16-33 User and Group tab in the Netscape console

6. Go to the drop-down menu situated in the lower-right corner of the Users and Groups tab and select **New User**. Click the **Create...** button. A dialog box is displayed, used to select the directory subtree in which to create the new entry, as shown in Figure 16-34.

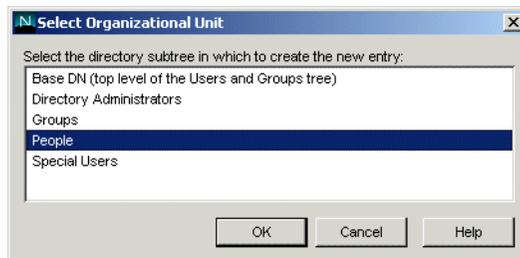


Figure 16-34 Select Organizational Unit window

7. Select **People** and click **OK**; the **Create User** window will appear (see Figure 16-35).

Figure 16-35 Creating a new user in the Netscape console

8. The fields with a * (asterisk) are required fields. Enter the First Name, Last name, Common Name and User ID. Include a Password for the wasadmin user and click **OK**.

Notice that the user ID will be used as the Security Server ID for configuring the Global Security Settings in WebSphere.

9. To view this new entry in the Directory, it is possible to search the user by including a unique string that can be found in the directory in the *Search, Users, Groups and Organizational Units for* field, as shown on Figure 16-36. Double-clicking the selected user allows you to modify the properties.

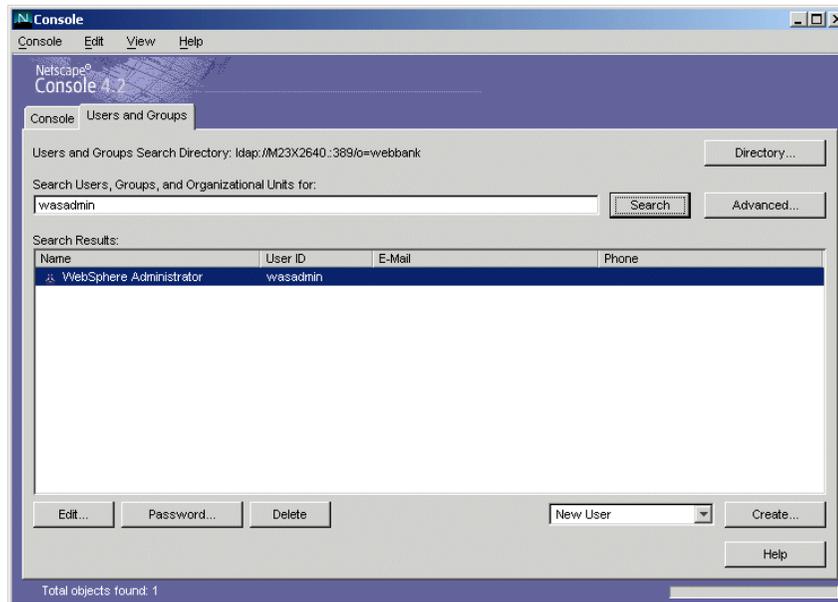


Figure 16-36 Searching wasadmin in the Netscape console

10. To view the contents of the LDAP directory, go to the Console tab and select the Directory Server in the left pane, then click **Open** in the right pane, as shown in Figure 16-37.

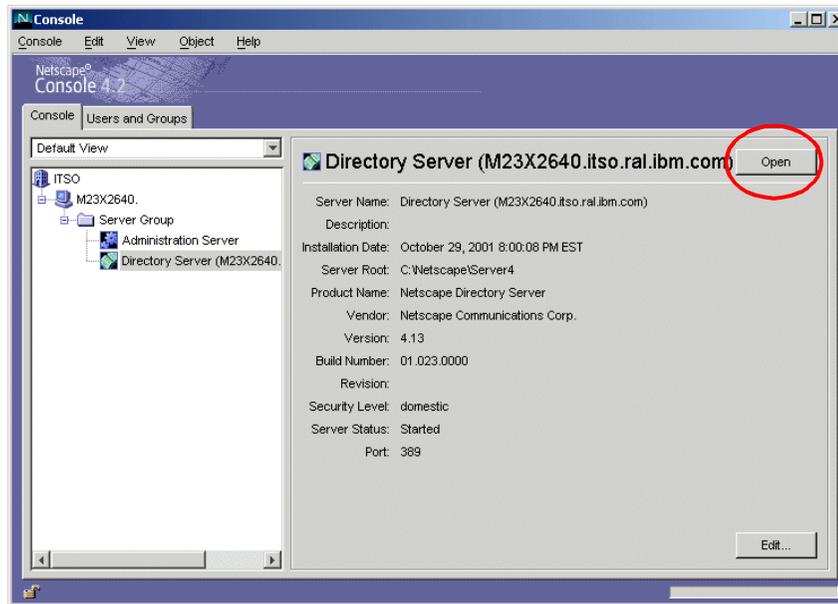


Figure 16-37 Opening the Directory Server in Netscape console

11. This button will bring up a new window; switch to the **Directory** tab and expand the **Webbank** folder to see its contents. Select **People** and the new WebSphere administrator user will appear, as shown in Figure 16-38

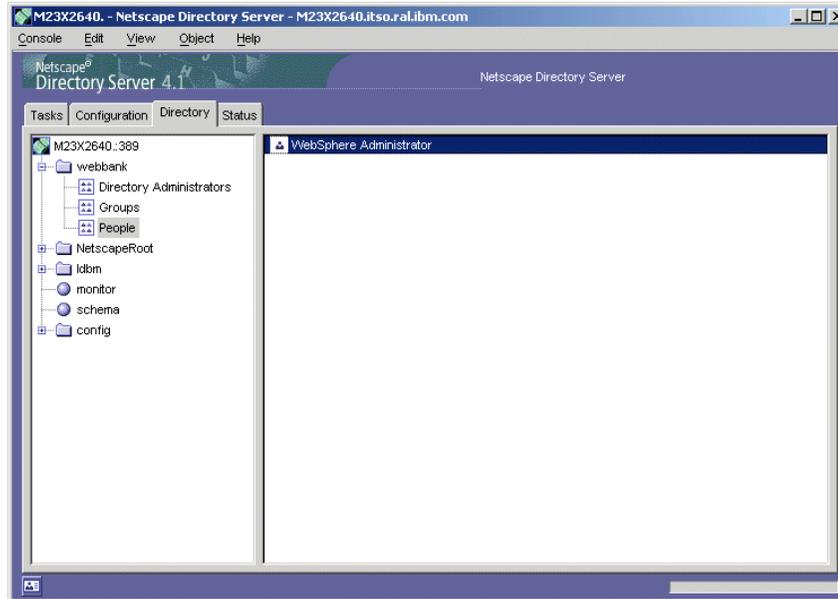


Figure 16-38 Netscape Directory Server Tree

16.3.2 Configuring WebSphere to use the Netscape Directory Server

To configure WebSphere to use Netscape Directory Server as its User Registry, follow these steps:

1. Start the WebSphere Administrator's Console (WebSphere Admin Server 4.0 must be running).
2. Select **Console** -> **Security Center...** This will display the Global Security settings for WebSphere. Select **Enable security** under the General tab.
3. Click the **Authentication** tab and specify the following settings:
 - Lightweight Third Part Authentication (LTPA).
 - Select the **LDAP** radio button. This will display the LDAP settings. Set the fields as follows:
 - **Security Server ID:** this field should contain the value specified in the User ID field for the WebSphere administrator user created previously.
 - **Security Server Password:** enter the password set for the wasadmin user in the Create User panel.
 - **Host:** this is the Host ID (IP address or DNS name) for the Netscape Directory Server.
 - **Directory Type:** select **Netscape** as the LDAP server.

- **Port:** the port number, by default, is 389 if none is specified. If the Default Netscape LDAP port number changes, it will be necessary to specify it in this field.
- **Base Distinguished Name:** this is specified in the User Directory subtree that appears under the Console tab, showing the Administration Domain information. This field is required for Netscape Directory Server.
- **Bind Distinguished Name:** the distinguished name used when WebSphere binds with the Netscape server. If no name is specified, the administration server will bind anonymously.
- **Bind password:** if a user is specified in the Bind Distinguished Name field, the password is required here.

4. All the settings are illustrated in Figure 16-39.

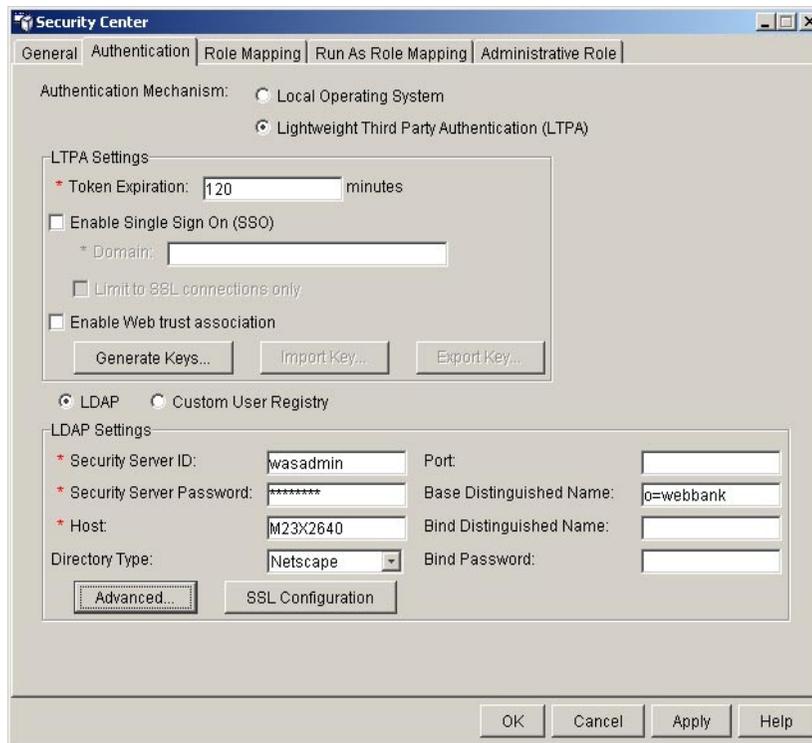


Figure 16-39 LDAP settings for Netscape Directory Server In WebSphere security

Click the **Advanced** button; a new window appears showing the LDAP advanced properties, as shown in Figure 16-40. There you can see how WebSphere will query the Netscape Directory.

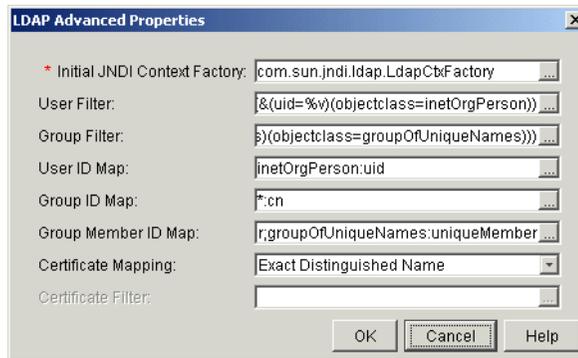


Figure 16-40 LDAP Advanced Properties window for Netscape

The default Netscape filters in LDAP Advanced Properties will perform the following tasks:

The User Filter will find a user entry in the Netscape Directory with the attribute *uid*, belonging to the *inetOrgPerson* object class, set with the name entered at authentication. It is typically used for the assignment of a Security Role to a User.

The Group Filter will find the group entries in the Netscape Directory with the attribute *cn*, belonging to the *groupOfNames* or *groupOfUniqueNames* object classes, set with the group name you are looking for. It is typically used for the assignment of a Security Role to a Group.

Note: *inetOrgPerson*, *groupOfNames* and *groupOfUniqueNames* object classes are part of the Netscape schema categories; to find out more about the Netscape Directory schema, refer to the *Netscape Directory Server Schema Reference Guide*.

The User ID map and Group ID map are the filters that map the short name of a User or Group to an LDAP entry; this information represents Users or Groups when they are displayed. LDAP returns the field specified by the User ID map and Group ID map (*uid* in the case of a user and Common Name in the case of groups) when the search performed by WebSphere is successful. WebSphere uses these returned values for comparison with permission lists.

The Group Member ID map identifies membership of Users to Groups.

These default settings will be suitable for searching Users and Groups in Netscape Directory Server. When the LDAP advanced properties are modified, the Directory Type changes to **Custom** in the Security Center window; this reflects the fact that the default settings for searching are no longer being used.

5. Click **OK**. The Security Server ID and Security Server Password will be verified against the information in the Netscape Server.

Make sure that the Netscape Directory Server is running by checking the list of Windows' services in the Netscape Directory machine and pinging the IP address of the Netscape Directory machine from the WebSphere machine.

The first time security is enabled, it will prompt for an LTPA password.

The LTPA password is used to protect the set of encryption keys generated for use in a Single Sign-On environment. For more information about Single Sign-On, refer to Chapter 14, "Single Sign-On" on page 393.

When the process is complete, a warning message will be displayed, stating that changes will not take effect until the Administration Server is restarted.

6. Restart the Administration Server by selecting the node, then right-clicking it and selecting **Restart** in the resulting context menu.
7. When the server is open for e-business, start the Administrator's Console. You will be prompted for the administrator user ID (Security Server ID) and password (Security Server Password).

16.4 Microsoft Active Directory

Active Directory is the directory service for Windows 2000 and is included in the operating system as part of the Windows 2000 server platform. It stores information about objects on the network, such as user accounts, shared printers, and other network objects, and provides access to this information.

This section explains how to configure WebSphere to use the Microsoft Active Directory as its LDAP server; to do this:

1. Add a new administration user in the Microsoft Active Directory for use with WebSphere.
2. Configure WebSphere to use the Microsoft Active Directory as its user registry.

16.4.1 Adding a new user

The following steps will guide you through the process of creating a new user in the Microsoft Active Directory.

1. Install and Configure the Active Directory server by clicking **Start -> Programs -> Administrative Tools -> Configure your Server**. The Configure Server window will be displayed. Select **Active Directory** and follow the instructions of the active directory wizard.

Note: The active directory wizard will install and configure the server as a domain controller and will set up DNS if it is not already available on the network.

- Before adding a new administrative user account for WebSphere, create a new organizational unit (OU) in the top level directory by selecting **Start -> Programs -> Administrative Tools -> Active Directory Users and Computers**. This will bring up the Active Directory Administration window, as shown in Figure 16-41.

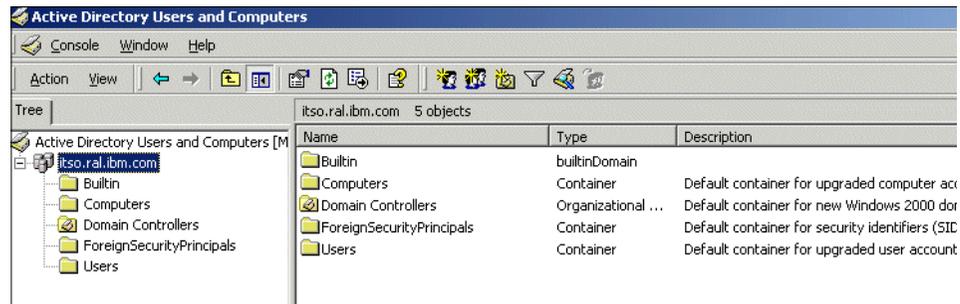


Figure 16-41 Active Directory Users and Computers window

- Select the **Domain** node in the left panel and right-click it. Select **New -> Organizational Unit**, as shown in Figure 16-42.

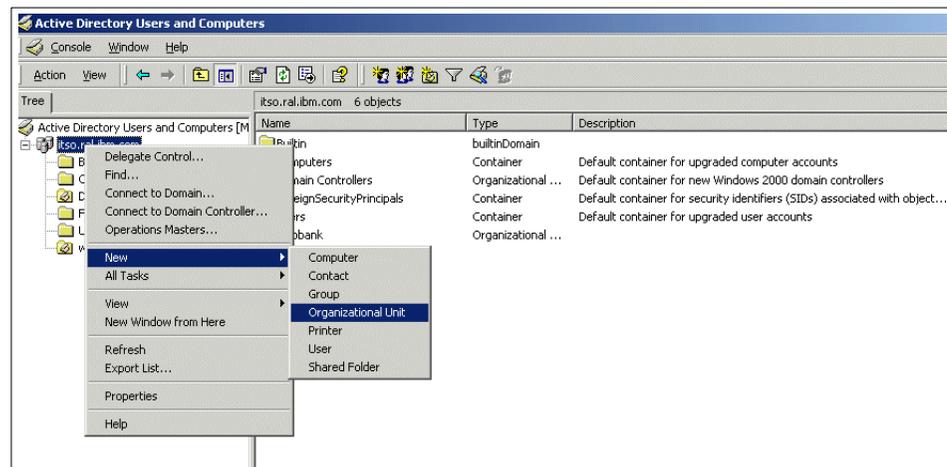


Figure 16-42 Adding a new organizational unit

- Specify a name for the new organizational unit (webbank) in the New Object -> Organizational Unit window **Name** field, as shown in Figure 16-43.

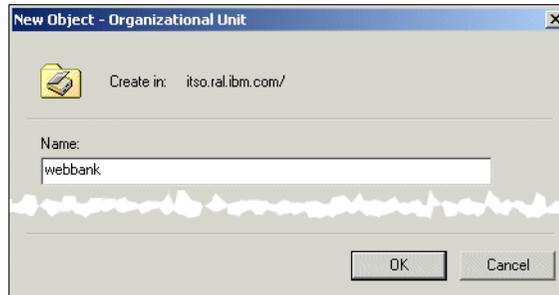


Figure 16-43 New organizational unit object

- Click **OK**. The new organizational unit appears in the Users and Groups Administration window.
- Now we are ready to add new users to the new organization created previously.

Select the **Webbank Organizational Unit** in the left pane of the console tree and right-click it. Select **New -> User**, as shown in Figure 16-44.

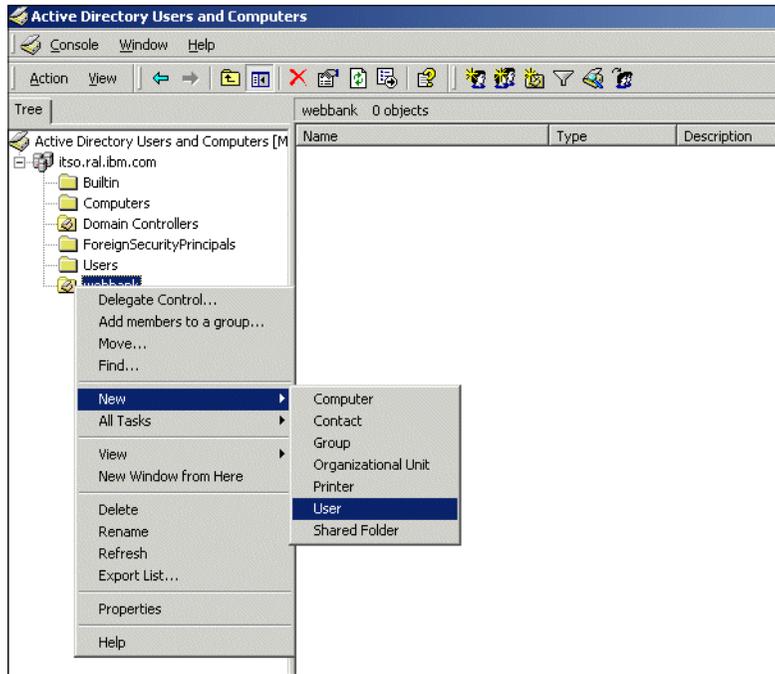


Figure 16-44 Adding a new user

7. Type the user's First Name, Last name and Full Name. Type the name with which the user will log on in the User logon name field. Notice that the User logon name will be used as the Security Server ID for configuring the Global Security settings in WebSphere.
8. From the drop-down menu, click the UPN (User Principal Name) suffix that must be appended to the user logon name (following the @ symbol). By default, the NS Domain name of the Domain contains the user account.

If the user uses a different name to log on from a computer running Windows NT, Windows 98, or Windows 95, change the user logon name as it appears (pre-Windows 2000) to the different name.

All these settings are illustrated in Figure 16-45.

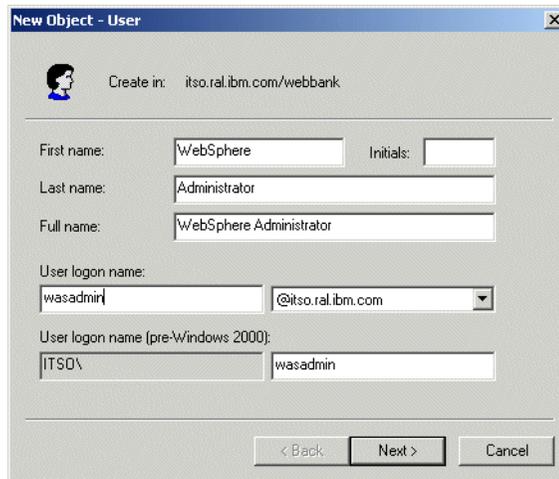


Figure 16-45 Creating a new user account

9. Click **Next** and, in the next window, type in the password for the wasadmin user.
10. Select **Password Never expires**.
11. Click **Next** and the next window will display the information for the new account to be created.
12. Click **Finish**.

After creating the user account, edit the user account properties to enter additional user account information.

To view the new user, refresh the administrative interface and highlight the organizational unit **webbank**, as shown in Figure 16-46.

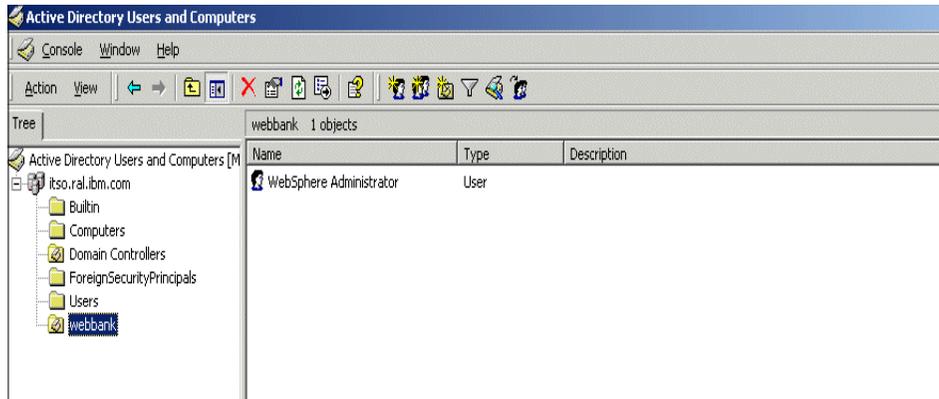


Figure 16-46 Showing the new WebSphere Administration user in Active Directory

16.4.2 Configuring WebSphere to use the Active Directory Server

To configure WebSphere to use Microsoft Active Directory Server as its User Registry, follow these next steps:

1. Start WebSphere Administrator's Console (the WebSphere Admin Server 4.0 must be running).
2. Select **Console** -> **Security Center...** This will display the Global Security settings for WebSphere. Select **Enable security** under the General tab.
3. Click the **Authentication** tab and specify the following settings:
 - Lightweight Third Part Authentication (LTPA).
 - Select the **LDAP** radio button; it will display the LDAP settings. Set the fields as follows:
 - **Security Server ID:** this field contains the value specified in the *User logon name* field for the WebSphere administrator user.
 - **Security Server Password:** enter the password set for the wasadmin user.
 - **Host:** the Host ID (IP address or DNS name) for the Microsoft Active Directory Server.
 - **Directory Type:** select **Active Directory** as the LDAP server.
 - **Port:** the port number, by default, is 389 if none is specified. If the Default Active Directory LDAP port number changes, it will be necessary to specify it in this field.
 - **Base Distinguished Name:** this is specified in the domain name of the Active Directory machine (itso.ral.ibm.com), in our case.

- **DC=itso,DC=ral,DC=ibm,DC=com:** this field is required for the Active Directory because it indicates the starting point for LDAP searches of the Directory Service.
- **Bind Distinguished Name:** type a distinguished name used when WebSphere binds with the Active Directory server (CN=WebSphere Administrator,OU=Webbank,DC=itso,DC=ral,DC=ibm,DC=com).

Important: This field is required for Active Directory installed under Windows 2000 platforms, because by default Windows 2000 does not allow you to view group membership and other user and group information using anonymous access. However, Windows NT 4.0 does allow this degree of access.

- **Bind password:** include the Bind Distinguished Name's password here.

All the settings are illustrated in Figure 16-47.

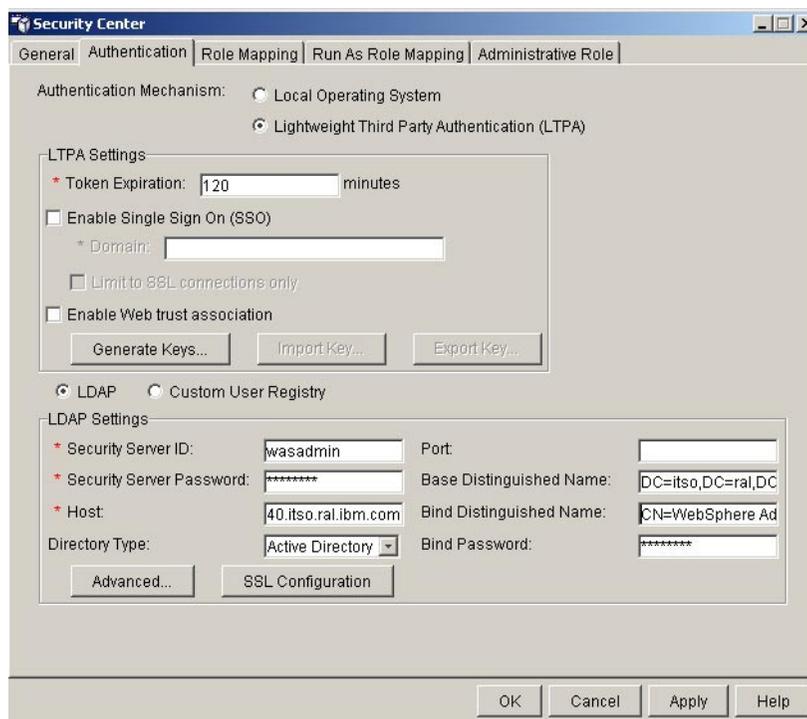


Figure 16-47 LDAP settings for Active Directory Server In WebSphere Security

These default settings will be suitable for searching Users and Groups in the Active Directory Server. When the LDAP advanced properties are modified, the Directory Type changes to **Custom** in the Security Center panel; this reflects the fact that the default settings for searching are no longer being used.

5. Click **OK**. The Security Server ID and Security Server Password will be verified against the information in the Active Directory Server.

Make sure that the Active Directory Server is running in the machine and pinging the IP address of the Active Directory machine from the WebSphere machine.

The first time security is enabled, it will prompt for an LTPA password.

The LTPA password is used to protect the set of encryption keys generated for use in a Single Sign-On environment; for more information, refer to Chapter 14, "Single Sign-On" on page 393.

When the process is complete, a warning message will be displayed.

6. Restart the Administration Server by selecting the node, then right-clicking it and selecting **Restart** in the resulting context menu.
7. When the server is open for e-business, start the Administrator's Console. You will be prompted for the administrator user ID (Security Server ID) and password (Security Server Password).



Using OpenSSL

This appendix provides a short example of how to set up your own CA for testing purposes using OpenSSL.

17.1 Open Source Software

Open Source Software (OSS) is a form of software distribution in terms of software licensing. It means, in a nutshell, that the software is freely distributable, that the program includes the source code, provides free licensing, and allows the developers to change the original code. There is more information about the Open Source initiative in the following Web site:

http://www.opensource.org/docs/definition_plain.html.

Aside from legal terms and licensing, Open Source Software also represents a free-minded and very effective software development methodology. The best example of this kind of development is the operatingsystem Linux.

17.2 OpenSSL

OpenSSL (Open Source Software) is freely available from <http://www.openssl.org/>. The program has a runnable part, which is a shell where users can manage all security-related procedures. It also includes APIs for development.

OpenSSL is based on security and Internet standards. The software is available as source code and binaries are also available on the Internet for different platforms.

Linux platform

OpenSSL runs perfectly under Linux, and is shipped with Linux products (for example Red Hat).

It works either from the shell prompt, using the **openssl** command or by running as a shell.

WIN32 platform

OpenSSL was originally developed for Unix systems, and since the source code is available, it is ported to WIN32, more or less. The program runs under WIN32, but the implementation still uses the original Unix resources and system calls. In order to run programs such as OpenSSL, where the code runs under WIN32 but uses Unix system resources and calls, shell software is required to provide the necessary environment for these types of applications.

The shell program is called *CygWin*, and is freely available from <http://www.cygwin.com>.

Setting up the directories

OpenSSL does not require any special directory to run. Since the results are files we will create here, it is always better to organize them in a directory structure.

The following directory names are only recommendations; you may use any other directories for OpenSSL.

We need only one directory for the procedure described next, to store all the result files and the entire CA. Create a directory called *openssl* under the /home directory. Before you run **openssl**, switch to this directory (/home/openssl), and you will see the results written there.

The directory is important in that the CA has to have certain files and a certain directory structure in order to run. This directory may be anywhere, since OpenSSL will use relative paths to find the CA, in case you set up several CAs under different directories. In our case, the working directory for OpenSSL will be /home/openssl.

17.3 How to create certificates using OpenSSL

In some cases, developers may want to have their own Certificate Authority (CA) in order to issue test certificates when self-signed certificates are not applicable. OpenSSL provides an easy way to manage your own CA, managing everything with only one command (**ca**) and its specific parameters.

For more information about the OpenSSL, see the documentation found at: <http://www.openssl.org/docs/>.

17.3.1 Creating your own CA

The following steps will guide you through the process of creating your own CA.

1. Open the openssl.cnf file in a text editor; OpenSSL has its file under the /usr/ssl directory.
2. Find the [CA_default] section in the configuration file.
3. The first directive is *dir*, which sets the CA directory for OpenSSL; the original value is *./demoCA*, you must change it to **./testCA**. This is a relative directory, which means that OpenSSL will look for the CA directory relative to the directory where it is executed. With this setting, you can have several different CAs set up without modifying the configuration file for OpenSSL.
4. Save the file, then close it.
5. Under the /home/openssl directory, create the directory **CA**.

6. Create the following subdirectories under CA: **newcerts**, **private**.
7. Create the file **index.txt**. It is simply an empty text file, and it will keep tracking the CA's activities. Use the command **echo >index.txt** to do so.
8. Create the file **serial**, with the content 0002. This file will keep tracking the serial number for the certificates. Use the command **echo 0002 >serial** to do so.

The directory and file structure are now set for the CA.

9. Create the certificate for the CA by issuing the following command:

```
openssl req -x509 -newkey rsa:1024 -keyout private/akey.pem -out cacert.pem
```

You will be asked for the password for the key file; enter the PEM pass phrase. Use your password; here, we will use `password` as our example. Type in the password again for verification.

The following fields have to be specified for the certificate:

- a. Country Name (by two-letter code) [AU]: US
- b. State or Province Name (full name) [Some-State]: North Carolina
- c. Locality Name (for example, your city) []: Raleigh
- d. Organization Name (for example, your company)[Internet Widgits Pty Ltd]:
IBM
- e. Organizational Unit Name (for example, your section) []: ITS0
- f. Common Name (for example, your name) []: testCA
- g. E-mail Address []: testCA@us.ibm.com

Now the CA is set and ready to work. You can test it by issuing the command **openssl ca** from the `/home/openssl` directory. You will be asked for the password, which is `password` in our example. If no error occurs, then the CA is working properly.

You may want to use the CA certificate to export the client certificate in PKCS#12 format, using the following command:

```
openssl pkcs12 -export -in cacert.pem -out cacert.p12 -name "testCA" -inkey private/akey.pem
```

1. First, you will be asked for the PEM password; use `password`.
2. Then the PKCS#12 password is required; type in `password` for example. You need to type this in twice for verification.

The *cacert.p12* is ready for import into any certification management tool which supports PKCS#12 format.

17.3.2 Client certificate

Following are directions to create the client certificate.

1. To create a certificate request, issue the following command:

```
openssl req -newkey rsa:1024 -keyout testclientkey.pem -out
testclientreq.pem
```

2. You will be asked for password; type in password for this sample. Then type in the password again for verification.

The following fields have to be specified for the certificate:

- a. Country Name (by two-letter code): US
- b. State or Province Name (full name): North Carolina
- c. Locality Name (for example, your city): Raleigh
- d. Organization Name (for example, your company): IBM
- e. Organizational Unit Name (for example, your section): ITS0
- f. Common Name (for example, your name): testclient
- g. E-mail Address: testclient@us.ibm.com

Please enter the following extra attributes to be sent with your certificate request:

- h. A challenge password: password
 - i. An optional company name: CLIENT
3. The next step is for you to sign the certificate requested for the client, using the following command:

```
openssl ca -in testclientreq.pem -extensions v3_ca -out testclientcert.pem
```

You will be asked for the certificate password; type in password for this example. The following information will be displayed:

Example 17-1 Verifying the certificate before signing

Check that the request matches the signature

Signature ok

The Subjects Distinguished Name is as follows

```
countryName          :PRINTABLE:'US'
stateOrProvinceName  :PRINTABLE:'North Carolina'
localityName         :PRINTABLE:'Raleigh'
organizationName     :PRINTABLE:'IBM'
organizationalUnitName:PRINTABLE:'ITS0'
commonName           :PRINTABLE:'testclient'
emailAddress         :IA5STRING:'testclient@us.ibm.com'
```

Certificate is to be certified until Nov 15 21:49:23 2002 GMT (365 days)

4. The CA will ask if you want to sign the certificate;, type in y.
5. Then you will be asked whether you want to commit the action; type in y.

OpenSSL is using its own format for files in order to exchange the certificates it has created; you can export them into the PKCS#12 format, which is widely accepted.

6. To export the client certificate in PKCS#12 format, use the following command:

```
openssl pkcs12 -export -in testclientcert.pem -out testclient.p12 -name
"test_client_certificate" -inkey testclientkey.pem
```

7. You will be asked for the PEM password; use password.
8. The PKCS#12 password is then required; use password for example. You need to type this in twice for verification.

17.3.3 Using the certificates

As a result, a couple of PEM and PKCS#12 files are generated with OpenSSL. The question is, how and where do we use them?

The PKCS#12 files are ready to import into IBM's ikeyman utility. Create either a JKS file or a KDB file, then import the PKCS#12 file as a personal certificate. PKCS#12 files store not only the certificate but the public and the private key pairs as well.

The format OpenSSL uses to store keys and certificates is called *privacy enhanced mail* (PEM), and is very like that of the files used for certificates: BASE64-encoded data surrounded by header lines. The IBM key management utility uses the .arm extension for these files.

The PEM files can be imported as .arm files into IBM's ikeyman utility without any changes. PEM files can store certificates, private keys, public keys, and so on. With the IBM key management utility, you can for example import certificates in PEM format for signers.



A

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246520>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-520.

Using the Web material

The additional Web material that accompanies this redbook includes the following file:

SG246520.zip: a Webbank sample application with security enhancements.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	10 MB minimum
Operating system:	Windows 2000 or AIX
Processor:	500MHz or higher
Memory:	512 MB or higher

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Refer to Chapter 5, “The sample used in this book” on page 73 for instructions on how to import the sample application into the development environment, and how to install it in a runtime environment.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 519.

- ▶ *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176
- ▶ *Deploying Public Key Infrastructure*, SG24-5512
- ▶ *Working with Business Process Objects for Tivoli PKI*, SG24-6043
- ▶ *Web Services Wizardry with WebSphere Studio Application Developer*, SG24-6292
- ▶ *Access Integration Using WebSphere Portal Server*, SG24-6267
- ▶ *Mobile applications with WebSphere Everyplace Access Design and Development*, SG24-6259
- ▶ *Enterprise Security Management with Tivoli*, SG24-5520
- ▶ *Tivoli SecureWay Policy Director - Centrally Managing e-business Security*, SG24-6008
- ▶ *Applying the Patterns for e-business to Domino and WebSphere Scenarios*, SG24-6255
- ▶ *Self-Service Patterns using WebSphere Application Server V4.0*, SG24-6175
- ▶ *User-to-Business Pattern Using WebSphere Personalization Patterns for e-business Series*, SG24-6213
- ▶ *User-to-Business Patterns Using WebSphere Advanced and MQSI: Patterns for e-business Series*, SG24-6160
- ▶ *e-commerce Patterns for Building B2C Web Sites, Using IBM WebSphere Commerce Suite V5.1*, SG24-6180
- ▶ *LDAP Implementation Cookbook*, SG24-5110

Other resources

These publications are also relevant as further information sources:

- ▶ Jonathan Adams, George Galambos, Srinivas Koushik, Guru Vasudeva, *Strategy for Reuse*, ISBN 1931182027.
- ▶ A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996
- ▶ *GPK Reference Manual (GemPlus 2001)*

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ Tivoli's Web site
<http://www.tivoli.com>
- ▶ Entrust's Web site
<http://www.entrust.com>
- ▶ Baltimore's Web site
<http://www.baltimore.com>
- ▶ RSA's Web site
<http://www.rsa.com>
- ▶ VeriSign's Web site
<http://www.verisign.com>
- ▶ Gemplus's Web site
<http://www.gemplus.com>
- ▶ Datakey's Web site
<http://www.datakey.com>
- ▶ Sun's Java Web site, Servlet 2.2 specification
<http://java.sun.com/products/servlet/2.2>
- ▶ Sun's Java Web site, EJB specification
<http://java.sun.com/products/ejb/docs.html>
- ▶ IBM WebSphere Advanced Edition V4.0 Infocenter
<http://www-3.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/index.html>

- ▶ IBM Software, WebSphere
<http://www.ibm.com/websphere>
- ▶ IBM Patterns for e-business Web site
<http://www-106.ibm.com/developerworks/patterns>
- ▶ Apache Web server's tutorials
<http://apache-server.com/tutorials/>
- ▶ OMG's Web site
<http://www.omg.org>
- ▶ W3C's Web site
<http://www.w3.org/>
- ▶ Thawte's Web site
<http://www.thawte.com>
- ▶ Open Source Web site
http://www.opensource.org/docs/definition_plain.html
- ▶ OpenSSL's Web site
<http://www.openssl.org/>
- ▶ Cygwin's Web site
<http://www.cygwin.com>
- ▶ Fortify's Web site
<http://www.fortify.net>

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Abbreviations and acronyms

AAT	Application Assembly Tool	JCE	Java Cryptography Extension
AC	Administrator's Console	JDBC	Java Database Connectivity
ACL	Access Control List	JKS	Java Key Store
AE	Advanced Edition	JSP	JavaServer Page
API	Application Programmer's Interface	JVM	Java Virtual Machine
CA	Certificate Authority	KDB	Key Database, file extension for IBM's iKeyman
CN	Common Name	LDAP	Lightweight Directory Access Protocol
CRL	Certificate Revocation List	LDIF	LDAP Data Interchange Format
CSR	Certificate Signing Request	LTPA	Lightweight Third Party Authentication
DD	Deployment Descriptor	ORB	Object Request Broker
DMZ	De-Militarized Zone	OS	Operating System
DN	Distinguished Name	OU	Organizational Unit
DNS	Domain Name Server	PD	Policy Director
EAR	Enterprise Application Archive	PID	Process ID
EJB	Enterprise Java Bean	PKI	Public Key Infrastructure
GSO	Global Sign-On	QOP	Quality Of Protection
GUI	Graphical User Interface	RA	Registration Authority
HTML	Hypertext Markup Language	RFC	Request For Comments
HTTP	Hypertext Transfer Protocol	RMI	Remote Method Invocation
IBM	International Business Machines Corporation	RPC	Remote Procedure Call
IDE	Integrated Development Environment	RPSS	Reverse Proxy Security Server
IHS	IBM HTTP Server	SAS	Secure Association Service
IIOP	Internet Inter ORB Protocol	SOA	Service Oriented Architecture
IOR	Interoperable Object Reference	SOAP	Simple Object Access Protocol
IT	Information Technology	SPI	Service Provider Interfaces
ITSO	International Technical Support Organization	SSL	Secure Socket Layer
J2EE	Java 2 Enterprise Edition	SSO	Single Sign-On

UDDI	Universal Description, Discovery, Integration
URI	Unified Resource Identifier
URL	Unified Resource Locator
VPN	Virtual Private Network
WAR	Web Application Archive
WAS	WebSphere Application Server
WSAD	WebSphere Studio Application Developer
WSCP	WebSphere Control Program
WSDL	Web Services Description Language
XML	eXtensible Markup Language

Index

A

- Access Control 6
 - List 10
- Access integration patterns 334
- Accountability 6
- Action method 120
- Active Directory Server settings 504
- Admin Repository 64
- Admin Server
 - disabling security 64
 - security 63
- admin.config 446
- Administration 6
- Administration Server 446
- Administrativon Server
 - securing 235
- Administrator's Console 52
- All authenticated users 82
- Anonymous LDAP connection 456
- Applet client 156, 168
- Application Assembly Tool (AAT) 41
- Application design 334
- Application integration patterns 334
- Application patterns 336
- Application security 37
- Application server 90, 339
- application.xml 48
- Assurance 6
- Authentication 6, 8, 17
 - flow 382
 - mechanism 168
 - method 112
 - method configuration 113
- Authorization 10

B

- Basic authentication 46, 112
- Basic Runtime pattern 341
- Biometric authentication 8–9
- Business patterns 334

C

- CA creating 511
- Caller's security context 203
- Caller's security information 204
- Capability List 10
- Centralized security 349
- Certificate 9, 17
 - exact mapping 270
 - Practice Statement 25
 - Repository 20
 - Revocation List 20
 - signed by a third-party CA 224
- Certificate Authority (CA) 19, 217, 511
- Certificate Signing Request (CSR) 226, 242
- Certificates authentication flow 58
- Cipher Support Strength 248
- Client 11
 - authentication 47
 - identity 11, 44
- Client certificate
 - authentication 113
 - creating 513
- Client-side
 - certificate 253
 - login 161
 - login process 162
- Common Name 258
- common repository 41
- Communicator Certificate DB 268
- Composite patterns 334
- Confidential transport guarantee 119
- Confidentiality 6, 17
- CORBA security 66
- CORBA Security Service 158
- Creating
 - a Client Keyfile 223
 - a Client trust file 224
 - a sample application server 90
 - a self-signed certificate 219
 - a self-signed certificate key pair 221
 - a server keyfile 219
 - a server trust file 223
- Credential
 - mapping 344

- transformation 344
- Credential propagation 344
- Credentials 158
- Cross certification 24
- Custom Login 132
- Custom Registry
 - configuration 206
 - sample 60, 78
 - SPI 206
- Custom User Registry 59
 - authentication mechanism 239
- CygWin 510

D

- Data integrity 17
- Database Access Control List (ACL) 409
- Database node 340
- DB2 custom registry 206
- Default SSL Configuration 53, 230
- Delegation 11
- Delegation policy 44, 147, 149–150
- De-Militarized Zone (DMZ) 339
- Demo keyring 217
- Denial of service 354
- Deploy 89
- deployment descriptor 41
- Development environment 74
- digest 17
- Digest authentication 47
- Directory Assistance document 402
- Directory Management Tool 360
- Directory services 340
- Disabling Anonymous LDAP searches 294, 314
- Distinguished Name 258, 264
- Domain firewall 339
- Domino
 - Directory settings 484
 - LDAP
 - self-signed certificate 318
 - Server 482
 - Server configuration 482
 - SSL configuration 324
 - sample import 79
 - security settings 428
 - Webbank sample 98
- dominoGroup object class 488
- dominoPerson object class 488

E

- EJB
 - client 156, 164
 - collaborator 35
 - container 157
 - security methods 203
- ejb-jar.xml 48
- Embedded HTTP server 270
- Enable Single Sign On (SSO) 237
- Enable Web Trust Association 238
- Enabling Global Security 235
- Encryption 14
- Enterprise application node(s) 341
- Enterprise Java Bean (EJB) 136
- Envelope Editor 177
- ePerson object class 481
- Everyone 82
- Exchanging public certificate keys 322
- Exchanging public certificates 274
- Extended Single Sign-On Application pattern 338
- Extract public key 222

F

- Facade design pattern 12
- File Serving Servlet 117
- Finer grained security 202
- Form-based authentication 47, 113
- Form-based login 131

G

- Generate Certificate Signing Request 226
- Generating a client key file 229
- Generating a server key file 225
- getCallerPrincipal() 203
- getRemoteUser() 204
- getUserPrincipal() 205
- Global 37
- Global Default SSL configuration 218
- Global Security 216
- Global Security Kit (GSKit) 461
- Global Sign-On (GSO) 348
- groupOfNames object class 498
- groupOfNames object class 481
- groupOfUniqueNames object class 481, 498

H

- Hash algorithm 17

Heterogeneous Single Sign-On 343, 347
Homogeneous Single Sign-On 343, 346
htaccess 111
HTTP
 basic authentication 106
 Server authorization 111
 server task 409
 session support 408
 transport mechanism 278
 Transport Properties 279
HTTPS 216
 connections only 280

I

IBM HTTP Server 438
IBM ikeyman 219
IBM JSSE ikeyman 267
IBM SecureWay Directory
 adding a group 475
 adding a user 469
 configuration 464
 defining a suffix 468
 filter 481
 install 460
 populating 467
 Server 282, 460
IBM Tivoli SecureWay PKI 255
Identification 6
Identity assigned to a Specific Role 149
Identity of caller 149
Identity of EJB Server 149
IHS SSL support 244
IIOP with SSL 216
ikeyman 514
Import certificate reply from CA 227
Import trusted root certificate 226
inetOrgPerson object class 498
Install 89
Install personal certificate 256
Integral transport guarantee 119
Integration patterns 334
Integrity 6
Interoperable Object Reference 158, 523
iPlanet Directory Server 300
 CA-signed certificate 308
 certificate 301
 SSL configuration 300
 SSL support 306

isCallerInRole() 203
isUserInRole() 205
iv_password 63
iv_user 63

J

J2EE 156
 API 203
 application client 156
 security 67
Java command line keytool 224
Java Key Store (JKS) 179, 217
Java keytool 265
Java language security 66
Java Secure Sockets Extension (JSSE) 217
Java thin application client 156, 166
Java Virtual Machine (JVM) 448
JavaServer Pages (JSP) 124
JVM Trace Arguments 448

K

key generation 22
key-based authentication 8
Keytool list option 227
Knowledge based authentication 8

L

LDAP 57, 258
 advanced settings 258
 authentication 454
 Certificate Filter 259
 certificate filtering configuration 260
 Directory Schema 295
 realm 407
 restricting access 296, 298, 300
 self-signed certificate 284
LDAPS 216
ldapsearch 485
ldapxcfg 464
LDIF 477
LDIF import 478
Lightweight Third Party Authentication 253, 394
Limit to SSL connections only 238
Local operating system 54
Local Operating System Authentication mechanism 236
Log Analyzer 450

- Logical security 4
- Login facilities 131
- LogLevel 439
- Logout 132
- Lotus Domino LDAP SSL configuration 317
- Lotus Domino server 394, 482
- LTPA 344
 - authentication mechanisms 57
 - keys 369, 377
- LTPAToken 394
 - view 414

M

- MD5withRSA 225
- Message handler 177
- Method permission 137, 140, 143
- Methods mapping 137
- Microsoft Active Directory 499
 - add user 499
- Model-View-Controller 124
- Modifying uniqueidentifier field 261
- Monitoring 6
- MQSeries custom registry 206

N

- Netscape Directory Server 490
 - adding a user 490
 - settings 496
- Network sniffing 354
- New key database 219
- None transport guarantee 119
- Non-repudiation 6, 17

O

- Object Request Broker (ORB) 216
- Obtaining personal certificate 255
- Open Source Software (OSS) 510
- OpenSSL 510
- Operating environment 66
- Operating system security 66
- ORB 157

P

- Password encoding 68
 - tool 69
- Patterns for e-business 334
- PEM file format 514

- Performance 67
- Personal certificate 255
- Personal Certificate install 268
- Physical keys 9
- Physical security 4
- PKCS#12 514
- PKCS12 267
- PKI Policies 25
- Policy Director
 - associate ACL 364
 - create a group 363
 - create ACL 363
 - install 358
 - LTPA authentication 368
 - permissions 367
- Privacy 6
 - private key 15
- Product mappings 346
- Programmatic login 161
- Protect static pages 358
- Protect WebSphere URIs 365
- Protocol firewall 339
- public key 15
- Public key cryptography 15
- Public Key Infrastructure (PKI) 14, 253, 339

Q

- Quality of protection (QOP) 158

R

- Realm 128
- Redbooks Web site 519
 - Contact us xv
- Registration Authority 20
- Reverse Proxy Security Server (RPSS) 62, 213, 381
- RMI/IIOP 156
- Role mapping 93
- row-level security 202
- RPC router servlet 173
- Run-As
 - mapping 151
 - Mode 44, 149
- Runtime pattern variation 341
- Runtime patterns 338

S

- Sample data source 80, 90
- Sample LDAP settings 82
- SAS interceptor 157
- SAS properties 157, 159
- SAS settings 159, 161
- sas.client.props 234
- Search filter 375
- Secret key cryptography 14
- Secure Association Service (SAS) 35, 156, 216, 449
- Secure Socket Layer (SSL) 216
- Secure Sockets Layer (SSL) 16
- Secure Web services
 - certificates 197
 - logs 194
 - sample 183, 199
- SecureWay Directory for Domino 401
- SecureWay Directory Server settings 478
- SecureWay Directory Server SSL configuration 282
- SecureWay LDAP Directory SSL configuration 287
- Securing WebSphere - LDAP communication 282
- Security
 - architecture 34
 - constraint 42, 112, 118, 124
 - identity 148
 - mapping 96
 - policy 5
 - services 340
 - settings locations 39
- Security and Administration 336
- Security Cache Timeout 68
- Security Center 52
- Security Collaborators 35
- Security Management 6
- Security Policies 36
- Security role mapping 93, 95, 204
- Security role reference 43, 122, 145, 204
- Security roles 81, 83, 86, 112, 140
- Security Server 35
 - ID 453
- Server configuration document 483
- Server side authentication 164
- Server-side login 163
- Server-side login process 163
- SHA1withDSA 225
- show task command 489
- Signature 17

- Signature Header Handler 178
- Single Sign-On 48, 98, 337, 348, 394
- Single Sign-On Application pattern 337
- Single Sign-On Runtime pattern 342
- Smart cards 25
- SOAP 172
 - client 189
 - envelope 178, 180
 - message 178, 180
 - security 172
 - signature 175
- Source address spoofing 354
- Special subjects 45
- special subjects 82
- Specified identity 11, 44
- SSL
 - configuration httpd.conf 245
 - handshaking 442
 - settings 46
- SSL V2 Ciphers 248
- SSL V3 Ciphers 248
- SSLCipherSpec checking 249
- SSLClientAuth directive 440
- SSLV3Timeout property 68
- SSO configuration 400, 405, 418, 426
- SSO configuration SSL 420, 430
- SSO testing 416, 422, 429
- Stand-alone Java application 76
- Starting the sample 93
- Static resources 106, 117
- symmetric key cryptography 14
- Symptom database 450
- System identity 11, 44

T

- Testing client certificate 263
- Testing SSO 411
- Thawte Consulting 255
- Third-party Certificate Authority 255
- Tivoli Policy Director 350
- Tivoli SecureWay Policy Director 344
- Token Expiration 237
- Transport guarantee 119, 281
- Transport hook 175
- Troubleshooting 432
- Trust Association Interceptor 381
- Trust Association Interceptor SPI 213

U

User name and password 9

V

Verification Header Handler 180

virtual private networks 354

W

Web application

server node 340

using client certificates 254

Web collaborator 35

Web Container self-signed certificate 272

Web Container SSL support 277

Web module security 127

Web security proxy 340

Web Serve

redirector 339

Web Server

certificate generation 240

HTTPS configuration 239

LDAP authentication 107

plug-in 270

configuration 367

configuration file 275

LogLevel 442

self-signed certificate 271

trace 442

SSL support configuration 244

support client certificates 262

Trace 439

Web Service configuration files 191

Web Service Java Bean Identity 187

Web Service Test Client 189

Web service wizard 184

Web services 172

Web services Admin 192

Web SSO Configuration Document 409

Web trust Association 62

Web Trust Association enable 383

web.xml 48

Webbank 89

Webbank sample 74

Webbank sample import 77

WebSEAL 213

junction 362

junction SSL 383

Web server 360

WebSphere Advanced Edition V4 ptf2 70

WebSphere Application Server 344

Trace 451

WebSphere Commerce Suite 344

WebSphere Control Program (WSCP) 37

WebSphere keyring configuration 229

WebSphere LDAP configuration 478, 484, 496, 504

WebSphere LDAP key database 286

WebSphere LDAP SSL configuration 290, 310, 328

WebSphere security 67

WebSphere self-signed certificate 321

WebSphere Studio Application Developer 74

WebSphere Test Environment 79, 190

WebSphere trace settings 444

WebSphere Web Server plug-in 440

Windows NT domain-controller 54

X

X.509 18

XML Digital Signature 172

XML signature 175

XMLConfig SSL configuration 232

XML-SOAP Admin 192

Z

z/OS 70



Redbooks

IBM Websphere V4.0 Advanced Edition Security

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



IBM WebSphere V4.0 Advanced Edition Security



IBM WebSphere Application Server security in detail

End-to-end security using Tivoli Policy Director

Single Sign-On for application servers

This IBM Redbook provides IT Architects, IT Specialists, application designers, application developers, application deployers and consultants with information to design, develop and deploy secure e-business applications using WebSphere Application Server V4 Advanced Edition.

The book focuses on WebSphere Application Server's security features and services. It provides a detailed overview of how to administer WebSphere security and how to secure Web components, EJBs, Java clients and Web services.

It also provides details about end-to-end security solutions and application design. You will find an introduction to Patterns for e-business, in which security is in focus. This book will also address Tivoli Policy Director and how to use it with WebSphere Application Server. Single Sign-On between application servers such as WebSphere and Lotus Domino is discussed in detail.

Finally, this redbook will provide additional information on problem determination, installation and configuration of WebSphere and different LDAP servers, and how to use OpenSSL to run your own CA.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks