WebSphere. software

IBM

WebSphere Application Server V7.0 Security Guide

Secure WebSphere administration processes

Ensure secure WebSphere applications

Secure communication with SSL

Carla Sadtler
Fabio Albertoni
Leonard Blunt
Shu Guang Chen
Elisa Ferracane
Grzegorz Smolko
Joerg-Ulrich Veser
Sean Zhu

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere Application Server V7.0 Security Guide

June 2009

SG24-7660-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xiii.

First Edition (June 2009)

This edition applies to WebSphere Application Server V7.

Contact an IBM Software Services Sales Specialist

Start SMALL, Start BIG, ... **JUST START**

architectural knowledge, skills, research and development . . .

that's IBM Software Services for WebSphere.

Our highly skilled consultants make it easy for you to design, build, test and deploy solutions, helping you build a smarter and more efficient business. **Our worldwide network of services specialists wants you to have it all!** Implementation, migration, architecture and design services: IBM Software Services has the right fit for you. We also deliver just-in-time, customized workshops and education tailored for your business needs. You have the knowledge, now reach out to the experts who can help you extend and realize the value.

For a WebSphere services solution that fits your needs, contact an IBM Software Services Sales Specialist:
ibm.com/developerworks/websphere/services/contacts.html

Contents

Notices

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

- AIX®
- CICS®
- DB2®
- developerWorks®
- Domino®
- i5/OS®
- IBM®
- Lotus®
- RACF®
- Rational®
- RDN®
- Redbooks®
- Redbooks (logo) ®
- Tivoli®
- WebSphere®
- z/OS®
- z9®

The following terms are trademarks of other companies:

Interchange, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

EJB, Enterprise JavaBeans, J2EE, J2SE, Java, JavaBeans, JavaServer, JDBC, JDK, JMX, JSP, JVM, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Internet Explorer, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbooks® publication provides the information that is needed to implement secure solutions with WebSphere® Application Server V7.0. It focuses on security for the application server and its components, including enterprise applications.

This book includes administrative and infrastructure security, application security, and z/OS specifics.

This book is intended for anyone who plans to secure applications and the application serving environment.

The team that wrote this book

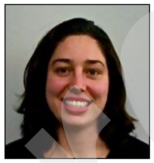
This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

Carla Sadtler is a Consulting IT Specialist at the ITSO, Raleigh Center. She writes extensively about WebSphere products and solutions. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative, supporting MVS clients. She holds a degree in mathematics from the University of North Carolina at Greensboro.

Fabio Albertoni is a Senior IT Specialist working in Integrated Technology Delivery SSO, on Hortolandia, Brazil. He has twelve years of experience in the IT and banking industries. He has spent the last eight years developing and implementing integrated solutions using WebSphere Application Server and MQ Series. He holds a degree in Data Processing from FATEC University of Ourinhos and a Masters degree in Computer Engineering from Instituto de Pesquisas Tecnologicas of Sao Paulo, Brazil.

Leonard Blunt is a Senior I/T Specialist working in ASEAN software lab services, based in Singapore. Leonard has a history in middleware architecture design and development, with emphasis on multi-channel e-business applications and application integrations. Leonard's origins are in building application middleware architectures, with a focus on rapid application development through product integration and the generation of code. Leonard is experienced in implementing J2EE/Java™ and service-oriented architecture (SOA) solutions and is passionate about producing robust hardened software that incorporates from its inception performance, monitoring, and security. Leonard has been working with WebSphere Application Server since 2003, and graduated from Wollongong University in New South Wales Australia with a Bachelor of Engineering (Computer) in 1999.

Shu Guang Chen is an Advisory Product Services Professional in IBM China. He has over nine years of experience in the IT field, and has worked as WebSphere technical support in IBM China for the last eight years, focusing on WebSphere Application Server and Portal administration, problem diagnostics, and performance tuning. He holds a degree in Computer Science and is certified in WebSphere Application Server V3.5, V4, V5, and V6.1, Java, AIX®, Portal, SOA, and DB2®.



Elisa Ferracane is a Software Developer working with the IBM WebSphere Security Development team in Austin, Texas. Her areas of expertise are WebSphere Security for z/OS® and Common Security Interoperability Version 2 (CSIv2). She has also worked as a system tester for WebSphere Application Server for z/OS. She has been with IBM for over seven years. Elisa received a Bachelor's degree in Computer Engineering from the University of Puerto Rico.

Grzegorz Smolko is a Certified IT Specialist with IBM Poland in Warsaw. Grzegorz has been working for IBM for six years in IBM Software Services for WebSphere. Prior to joining IBM, he worked for software house companies in Poland as a Java developer and architect. His areas of expertise include Java, Java Enterprise Edition (JEE), and WebSphere. He holds certifications from Sun™ and IBM in Java and WebSphere technologies. He has a Master's degree in Computer Science from the Warsaw University of Technology, Poland.

Joerg-Ulrich Veser is an IT Specialist working since 2006 in the pre-Sales support for WebSphere on z/OS in Germany. His areas of expertise include infrastructure architecture design, implementation, problem determination, high availability, and security on WebSphere products for z/OS. He holds a degree in Computer Science from the University of Cooperative Education in Mannheim (Germany).

Sean Zhu is an IBM Certified IT Specialist in the IBM Business Partner Technical Strategy & Enablement (BPTS&E) organization. He has over ten years of IT consulting experience, specializing on business process reengineering and integration using IBM SOA middleware and eCommerce solutions using WebSphere Commerce. He is experienced in integrating major ERP systems using WebSphere Adapters, Enterprise Service Bus (ESB) and WebSphere Process Server. He has published IBM Redbooks publications, white papers, and developerWorks® articles and has presented at conferences. He is certified in approximately twelve IBM products and technologies, including WebSphere Process Server, WebSphere Commerce, and WebSphere Application Server. He holds a Master of Science in Information Systems degree and a Master of Business Administration degree, both from Arizona State University. He is also a The Open Group (TOG) Master Certified IT Specialist.

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

Charles Lewis
NSI Technology

Bill O'Donnell
IBM US

Keys Botzum
IBM US

Emily Tuczkowski
IBM US

Vishwanath Venkataramappa
IBM US

Daniel Morris
IBM US

Don Bagwell
IBM US

Holger Wunderlich
IBM Germany

James Kochuba
IBM US

Thanks to the authors of the previous editions of this book, the authors of the *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316, which was published in December 2006: Rufus Credle, Tony Chen, Asish Kumar, James Walton, and Paul Winters

Become a published author

Join us for a two- to six-week residency program. Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, IBM Business Partners, and Clients.

Your efforts will help increase product acceptance and client satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

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

- Use the online **Contact us** review IBM Redbooks publications form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Part 1

Administrative and infrastructure security

1 Introduction

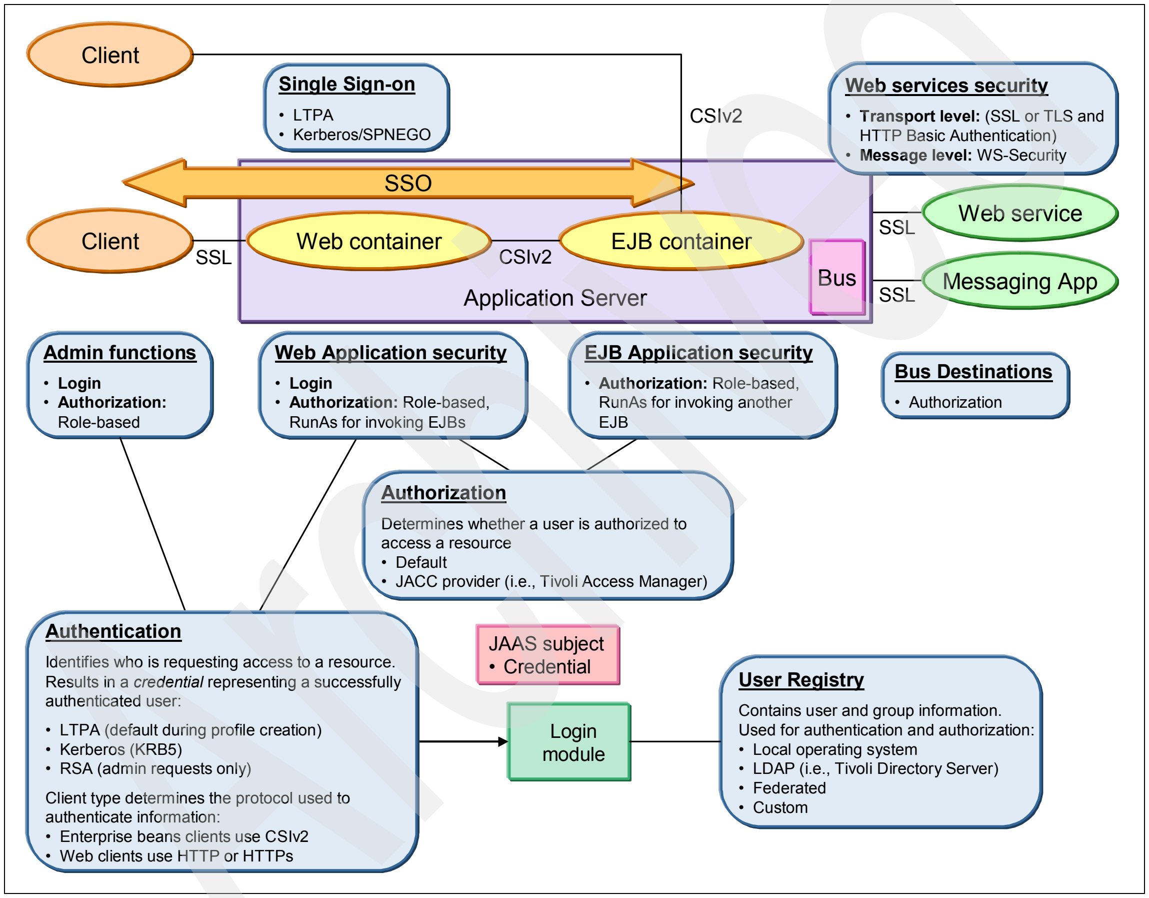
This chapter provides a short introduction to core security concepts and security technologies. We recommend that you understand the concepts that are documented here to help you better understand the later chapters. Subsequent chapters of this book will help you to understand how WebSphere Application Server implements, supports, or uses the concepts and technologies described here.

This chapter contains the following topics:

- “Core concepts and technologies” on page 4
- “Summary of new V7 security features and changes” on page 13

1.1 Core concepts and technologies

Securing a WebSphere Application Server environment can be complex and requires an understanding of your WebSphere and network infrastructure, the security requirements for the applications, and the users that you expect to access these applications. You must consider multiple aspects of security. Figure 1-1 shows a high-level view of a WebSphere Application Server environment and the points at which security can be implemented.



Client
Single Sign-on
• LTPA
• Kerberos/SPNEGO
CSIv2
Web services security
• Transport level: (SSL or TLS and HTTP Basic Authentication)
• Message level: WS-Security
SSO
Client
SSL
Web container
CSIv2
EJB container
Bus
Application Server
SSL
Web service
SSL
Messaging App
Admin functions
• Login
• Authorization: Role-based
Web Application security
• Login
• Authorization: Role-based, RunAs for invoking EJBs
EJB Application security
• Authorization: Role-based, RunAs for invoking another EJB
Bus Destinations
• Authorization
Authorization
Determines whether a user is authorized to access a resource
• Default
• JACC provider (i.e., Tivoli Access Manager)
Authentication
Identifies who is requesting access to a resource. Results in a credential representing a successfully authenticated user:
• LTPA (default during profile creation)
• Kerberos (KRB5)
• RSA (admin requests only)
Client type determines the protocol used to authenticate information:
• Enterprise beans clients use CSIv2
• Web clients use HTTP or HTTPs
JAAS subject
• Credential
Login module
User Registry
Contains user and group information. Used for authentication and authorization:
• Local operating system
• LDAP (i.e., Tivoli Directory Server)
• Federated
• Custom

Figure 1-1 High-level view of WebSphere Application Server security

1.1.1 Global security and security domains

WebSphere Application Server provides configuration facilities that allow you to secure the administrative applications and services that are used to manage and configure a WebSphere environment and to secure applications running in that environment. These configuration activities are done separately, although, they can share common settings.

Global security settings are the security configuration settings that apply to all administrative functions and provide the default settings for user applications.

New in V7, WebSphere Application Server V7 introduces the ability to create additional *security domains* to secure user applications and their resources. A security domain is specific to the application servers, clusters, and service integration buses that are assigned to it. A security domain can have attributes that differ from the global security settings. For example, a separate user registry can be used to secure administrative functions and applications.

You can also associate a security domain with the cell (referred to as a *cell domain*). In this case, the global security attributes are used to secure the administrative applications while the security domain attributes are used as the default for securing user applications. Additional security domains can be created and used for specific servers and clusters.

For more information: Refer to Chapter 3, "Using security domains" on page 51.

1.1.2 Securing the administrative environment

Administrative security provides the core of the security structure for WebSphere Application security and is the first step in configuring a secure system. You must activate administrative security before any other security configuration for the WebSphere processes can take effect. Administrative security settings are defined in the global security settings.

The primary function of administrative security is to prevent unauthorized access to the WebSphere processes. Administrative functions are secured based on predefined roles that have been assigned to users or groups. Each role carries a specific amount of authority over WebSphere runtime activity. For example, users that are assigned to a group that has the Monitor role can view the WebSphere configuration and status of the servers. Users that are assigned to a group that has the Operator role can also view the configuration and status, but in addition, can actually change the runtime state of processes (for example, starting or stopping an application server).

Administrative security is enabled for a cell, meaning that when you create a profile that configures a new cell (deployment manager profile, stand-alone application server profile, or cell profile option), you have the option to enable administrative security. This option is selected by default.

When you enable administrative security, you must configure the authentication mechanism and user registry to be used for authorization and authentication. This configuration is referred to as the *global security configuration* and applies to all administrative functions and to applications that run on servers that are not defined to another security domain.

Additional security domains can be configured that apply to user applications. When you create these additional domains, you can use a separate user registry than the user registry that is used for administrative security.

For more information:

- For more information about administrative security, refer to Chapter 2, "Administrative security" on page 17.
- For more information about security domains, refer to Chapter 3, "Using security domains" on page 51.

1.1.3 Defining user registries to WebSphere

WebSphere security relies on a user registry that contains information about users and groups for authentication and authorization of user IDs. A single registry might span multiple data repositories. The logical collection of users in a registry is referred to as a *user realm*. Only one registry or repository can be active for a security domain.

WebSphere Application Server supports four types of registries:

- The local operating system
- A federated repository
- A Lightweight Directory Access Protocol (LDAP) registry
- A custom registry

Be careful using the local operating system registry. Depending on the operating system and registry setup, you might end up using a registry that is not centralized. If you are running a WebSphere environment that is spread across multiple machines, each application server uses the operating system registry on the machine on which it runs, which, obviously, is not a wise choice. On z/OS however, the local user registry is a centralized registry within a sysplex. WebSphere uses the System Authorization Facility (SAF) interface to access the registry. SAF allows security authorization requests to be processed directly through the Resource Access Control Facility (RACF®) or a third-party z/OS security provider.

Using a federated repository allows you to use multiple repositories that are combined under a single security realm. The repositories can be file-based, LDAP, a sub-tree of an LDAP repository, or a database. When you enable administrative security using the profile creation wizard, this is the option that is used with a file-based repository holding a single administrative ID that you select. There are limitations when using a federated repository. Refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwim_limitations.html

WebSphere Application Server supports several Lightweight Directory Access Protocol (LDAP) registries. For information about the supported LDAP products, refer to the system requirements for your platform at:

http://www-01.ibm.com/support/docview.wss?uid=swg27006921

WebSphere Application Server also provides a plug-in to support any registry by using the custom registry feature. The custom registry feature enables you to configure any user registry that is not made available through the security configuration panels of the WebSphere Application Server.

Before enabling administrative security, you must determine what type of registry you will use, and if you are not using the default registry (federated repository), you will need to have the registry in place and predefine the administrator user ID and password to it.

For more information: Refer to Chapter 4, “Configuring the user registry and authentication settings” on page 65.

1.1.4 Authenticating clients

Authentication is the verification of identity (user ID and password, digital certificate, and so forth). The authentication process checks the information provided by a user and determines if the user has provided sufficient information for the identity to be accepted.

When you configure the authentication mechanism to be used in a cell, you define how security information will be exchanged between the client and WebSphere process. The authentication mechanism works with the user registry to verify the identity of the client and then creates a *credential* that represents the authenticated client. The abilities of the credential are determined by the configured authentication mechanism. Only a single active authentication mechanism can be configured within the cell.

The authentication methods available are:

- Lightweight Third Party Authentication (LTPA)

  LTPA is intended for application server environments that are distributed across multiple machines and machine environments. LTPA also provides the single sign-on (SSO) feature, allowing a user to authenticate only once for all WebSphere applications in a cell. There is no inherent SSO to resources outside the cell (WebSphere MQ, databases, and so forth)

- Kerberos (KRB5)

  The Kerberos authentication mechanism enables end-to-end SSO interoperability with other applications that support Kerberos authentication. A Java client can participate in the Kerberos SSO using the Kerberos credential, not the user and password, to authenticate to WebSphere Application Server.

  Java 2 Platform, Enterprise Edition (J2EE™), Web service, .NET, and Web browser clients that use the HTTP protocol can use the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) token to authenticate to the WebSphere Application Server and participate in SSO by using SPNEGO Web authentication.

- RSA token authentication mechanism

  RSA is used for the administrative agent only.

Simple WebSphere Authentication Mechanism (SWAM) has been deprecated and must not be used.

Authentication for Web applications

In the WebSphere Application Server security structure, clients that access Web and Enterprise JavaBeans™ (EJB™) applications must authenticate when they attempt to access a secure resource. Authentication takes place differently depending on the client type.

When a Web client requires initial authentication with WebSphere, the three following options for authentication are supported:

- Basic authentication in which WebSphere challenges the browser, asking for a user ID and password. The browser requests the information from the user and passes it to WebSphere for authentication.
- Form-based login in which WebSphere invokes a custom login page.
- Certificate from the Secure Sockets Layer (SSL) connection is mapped to the registry.

In each case, the authentication is performed using the user registry. A Lightweight Third Party Authentication (LTPA) token is generated in a cookie for use in subsequent requests.

Authentication for EJB applications

An EJB client can be a J2EE application client or Java 2 Platform, Standard Edition (J2SE™) application external to the application server, or an application component within the application server running in another Web or EJB container.

If the client already has an authentication token, that token is used for authentication. If not, and authentication is required, the client must send the credentials to be authenticated (there is no challenge from WebSphere). The certificate or password is validated against the user registry, and a Common Security Interoperability Version 2 (CSIv2) session is created between the client and server. The LTPA token flows across the CSIv2 session (not an LTPA cookie).

For more information: Authentication is discussed further in 4.4, "Authentication and authorization settings" on page 133.

1.1.5 Authorizing access to applications

When a protected Java EE resource is accessed, authorization occurs to determine if the *subject* is allowed to have access to the resource. Authorization controls are commonly linked to lists or groupings of user entities as a way to simplify the management of access controls.

Security constraints define the authorization required by a user to an application resource. The constraint specifies the resource and the role that a user has to have to access it.

A resource is a servlet or EJB. Secured servlets are identified by URL pattern (as opposed to a servlet or class name), while secured EJBs are identified by EJB name and class. Methods within the resource can be secured individually. For servlets, the methods can be GET, POST, and so forth. For EJBs, you can secure access to individual methods: business methods, home methods, and so forth.

The `RunAs` property can be set for servlets or EJBs that define under which user ID the code is acting, which affects the codes' interaction with other secured resources. A J2EE role defines the permissions of the users and groups that are defined to this role. Roles are specified in the application and are mapped to users and groups defined in the user registry during deployment in the application bindings, or using the WebSphere administrative tools.

The security structure for an application can be expressed in a declarative manner using entries in WAR, EJB, or EAR deployment descriptors or annotations within the code, or programmatically within the application using the security API. Using the programming API can give you more control over the authorization process by allowing it to be instance-based and to use complex rules in determining if the user is authorized.

For more information: Refer to:

- Chapter 7, “Application security” on page 253
- Chapter 8, “Securing a Web application” on page 267
- Chapter 9, “Securing an Enterprise JavaBeans application” on page 325

1.1.6 Authorization providers

WebSphere Application Server supports both a default authorization provider and an authorization provider that is based on the Java Authorization Contract for Containers (JACC) specification. The JACC-based authorization provider enables third-party security providers (for example, Tivoli® Access Manager) to handle the Java EE authorization.

For more information: Using JACC providers is not discussed in this book. For more information, refer to *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316.

1.1.7 Protecting file systems with Java 2 security

Java 2 security provides protection for the code. It prevents code from accessing other code, regardless of the user context. Java 2 security is managed using policy files. The defaults for Java 2 security are restrictive, so a complete understanding of your WebSphere applications and the security implications is essential before you enable this type of security.

1.1.8 Single sign-on

Single sign-on (SSO) is popular with users, because it allows them to authenticate only once, yet they can access multiple applications including WebSphere applications. SSO simplifies user ID management from a user and IT support standpoint. However, consider an SSO solution carefully, because integration might not be possible with all applications in an enterprise. SSO also creates a possible security risk from unattended workstations where a user has authenticated. And finally, with SSO there is one central point of authentication, making an attack by a malicious hacker an even more serious risk.

SSO can be implemented in multiple ways, including the use of external devices (smart cards, for example) or through software implementations. WebSphere Application Server provides support for the use of LTPA cookies and Simple and Protected GSS-API Negotiation (SPNEGO). LTPA cookies do not require any particular client and allow SSO across various cells as long as the user registry and the LTPA keys are the same. SPNEGO uses the token from a Kerberos login (typically, Windows®) to authenticate to WebSphere Application Server.

New in V7, the trust association interceptor (TAI) that uses the SPNEGO to securely negotiate and authenticate HTTP requests for secured resources (introduced in WebSphere Application Server Version 6.1) is now deprecated with V7. SPNEGO Web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

1.1.9 Web services security

The Organization for the Advancement of Structured Information Standards (OASIS) Web services security (WS-Security) specification defines the core facilities for protecting the integrity and confidentiality of a message and provides mechanisms for associating security-related claims with the message. Web services security is a message-level standard based on securing SOAP messages through XML digital signature, confidentiality through XML encryption, and credential propagation through security tokens. WebSphere Application Server V7 supports Version 1.1 of the WS-Security specification, including features such as encrypted header, thumbprint and signature configuration, username token profile, and X.509 token profile. In addition, limited security scenario support is provided for the Kerberos Version 1.1 token profile, WS-SecureConversation Version 1.3, WS-Trust Version 1.3, and WS-SecurityPolicy Version 1.2.

For more information: Refer to *WebSphere Application Server V7 Web Services Guide*, SG24-7758.

1.1.10 Messaging security

WebSphere Application Server supports the following messaging providers:

- The WebSphere Application Server default messaging provider (which uses the service integration bus as the transport for the provider)
- The WebSphere MQ messaging provider (which uses your WebSphere MQ system as the provider). The WebSphere MQ messaging provider does not use service integration.
- Third-party messaging providers that implement either a Java Platform, Enterprise Edition Connector Architecture (JCA) Version 1.5 resource adapter or the application support filter (ASF) component of the JMS Version 1.0.2 specification.

With regard to the WebSphere Application Server environment, security for messaging using these providers can be defined at multiple points:

- A messaging client accesses a messaging provider by creating a connection to it. This connection can be secured by requiring authentication and authorization to take place for new connections. The credentials can be provided by the application or specified on the connection factory that is used to create the connection.
- Messages that travel over the network from the application server to the messaging destination can be protected by using SSL on the transport.
- Messages are stored on queue or topic destinations. These destinations can also be secured by requiring authentication and authorization to take place before storing or accessing messages on the destination.

Additional security points exist within WebSphere Application Server in a topology that uses the WebSphere default messaging provider. A service integration bus provides the underlying transport for this provider. Application servers and clusters are added as members of the bus, each having a messaging engine on the bus that provides the core messaging capabilities.

Communication between messaging engines can be secured by requiring authorization to take place. Messages stored on destinations in the bus can be stored on a file system or a database. If using a database, it can be protected, and a J2EE Connector architecture (J2C) authentication alias can be used to provide the credentials required for access.

For more information about securing the default messaging provider: Refer to *WebSphere Application Server V7 Messaging Administration Guide,* SG24-7770.

There are alternatives to connecting to a WebSphere MQ network using the WebSphere MQ messaging provider:

- A WebSphere MQ network can be defined as a foreign bus (using WebSphere MQ links). A *WebSphere MQ link* provides a server to server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network.

  Role-based authorization can be used to secure access to both the local bus and the foreign bus. You can also authorize users to access the foreign or alias destinations that will forward messages to a foreign bus.

- A WebSphere MQ server (a queue manager or queue-sharing group) provides a direct client connection between a service integration bus and queues on a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group.

  A WebSphere MQ server definition provides authentication settings that service integration uses to connect to the associated WebSphere MQ queue manager or queue-sharing group.

In each case, SSL can be used to secure communications between WebSphere Application Server and WebSphere MQ.

1.2 Summary of new V7 security features and changes

These new features and changes are relevant to security in WebSphere Application Server V7:

- Auditing the security infrastructure

  The new security auditing feature provides the infrastructure that allows you to implement your code to capture and store supported auditable security events. During run time, all code other than the Java EE 5 application code is considered to be trusted. Each time that a Java EE 5 application accesses a secured resource, any internal application server process with an audit point included can be recorded as an auditable event.

- Authorization providers

  The Java Authorization Contract for Containers (JACC) specification 1.4 is included to support Java EE 5, including the use of annotations for propagating security policy information.

- Configuring the Kerberos token for Web services security

  The support for Kerberos with Web services security in WebSphere Application Server Version 7.0 is included and is based on the OASIS Web Service Security Kerberos Token Profile 1.1 specification.

- General JAX-WS default bindings for Web services security

  The configuration of the default cell level and default server level bindings has changed in WebSphere Application Server Version 7.0. Previously, you configured only one set of default bindings for the cell and, optionally, configured one set of default bindings for each server. In Version 7.0, you can configure one or more general provider bindings and one or more general client bindings. However, only one general provider binding and one general client binding can be designated as the default.

- Kerberos (KRB5) authentication mechanism support for security

  Security support for Kerberos as the authentication mechanism has been added for this release of WebSphere Application Server. Kerberos is a mature, flexible, open, and extremely secure network authentication protocol. Kerberos includes authentication, mutual authentication, message integrity and confidentiality, and delegation features. You can enable Kerberos on the server side. Support is provided to enable the rich Java client to use the Kerberos token for authentication to the WebSphere Application Server. However, the thin client, administrative thin client, and application client do not support a Kerberos token for authentication to the Application Server. These thin clients must instead use BasicAuth/GSSUP for authentication to the Application Server.

- Multiple security domains

  Multiple security domain support allows you to create multiple security configurations and assign them to various applications in WebSphere Application Server processes. By creating multiple security domains, you can configure different security attributes for administrative and user applications within a cell environment. You can configure separate applications to use separate security configurations by assigning the servers, clusters, or service integration buses that host these applications to the security domains. Only users assigned to the administrator role can configure multiple security domains.

- Securing communications

  WebSphere Application Server provides several methods for securing communication between a server and a client. New in this release are functions that ensure secure communication between a server and a client. These functions focus on certificate management, authentication, and ensuring trust among the application server, administrative agent, and job manager. The new functions include:

  – Creating and using certificate authority (CA) clients to enable a CA to request, query, and revoke certificates
  – Creating and using chained personal certificates to allow a certificate to be signed with a longer life span
  – Creating and revoking certificate authority (CA) certificates to ensure secure communication between the CA client and the CA server
  – Allowing the WebSphere Application Server administrator to create, configure, and enable System Authorization Facility (SAF) keyrings by utilizing the Open Cryptographic Services Facility (OCSF) Data library functions for SAF keyrings

- Single sign-on for HTTP requests using SPNEGO Web authentication

  In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO Web authentication has taken its place to provide the following enhancements:

  – You can configure and enable SPNEGO Web authentication and filters on WebSphere Application Server by using the administrative console.
  – Dynamic reload of SPNEGO is provided without the need to stop and restart WebSphere Application Server.
  – Fallback to an application login method is provided if the SPNEGO Web authentication fails.
  – SPNEGO can be customized at the WebSphere security domain level.

2

Administrative security

Administrative security is used to secure access to the administrative functions for the WebSphere environment and to provide the basis for the WebSphere security infrastructure. Enabling administrative security activates a wide variety of security settings for WebSphere Application Server, including the mechanism used for authentication of users, the use of Secure Sockets Layer (SSL), and the choice of user account repository.

This chapter provides information about enabling and managing administrative security. It contains the following topics:

2.1 Administrative security overview

The term *global security* refers to the security configuration that applies to all administrative functions and that provides the default security configuration for user applications. By default, all administrative and user applications in WebSphere Application Server use the global security configuration. You can create additional WebSphere security domains if you want to specify separate security attributes for several or all of your user applications. Administrative security must be enabled before you can activate application security.

Enabling administrative security provides the ability to secure the basic infrastructure of a WebSphere environment, which includes:

- Authentication of HTTP and Internet Inter-ORB Protocol (IIOP) clients
- Security for administrative functions (administrative console access, access to execute commands, and access for **`wsadmin`** scripts)
- Naming service security
- Use of SSL transports

Enabling administrative security will require that you select a user registry. When you enable security during profile configuration, you automatically get a federated repository configuration. To select any other option, you will need to deselect the option to enable administrative security and enable it after the profile is created. The user registry that you select in the global security settings is used for administrative security and, by default, for application security. However, a separate user registry can be configured in a security domain for applications. We discuss user registries further in Chapter 4, “Configuring the user registry and authentication settings” on page 65.

2.2 Enabling administrative security

There might be environments where no security is needed, for example, on individual test systems that are used by developers. On these systems, you can elect to disable administrative security. However, in most environments, administrative security needs to be enabled to prevent unauthorized users from accessing the administrative functions.

2.2.1 Enabling security at profile creation

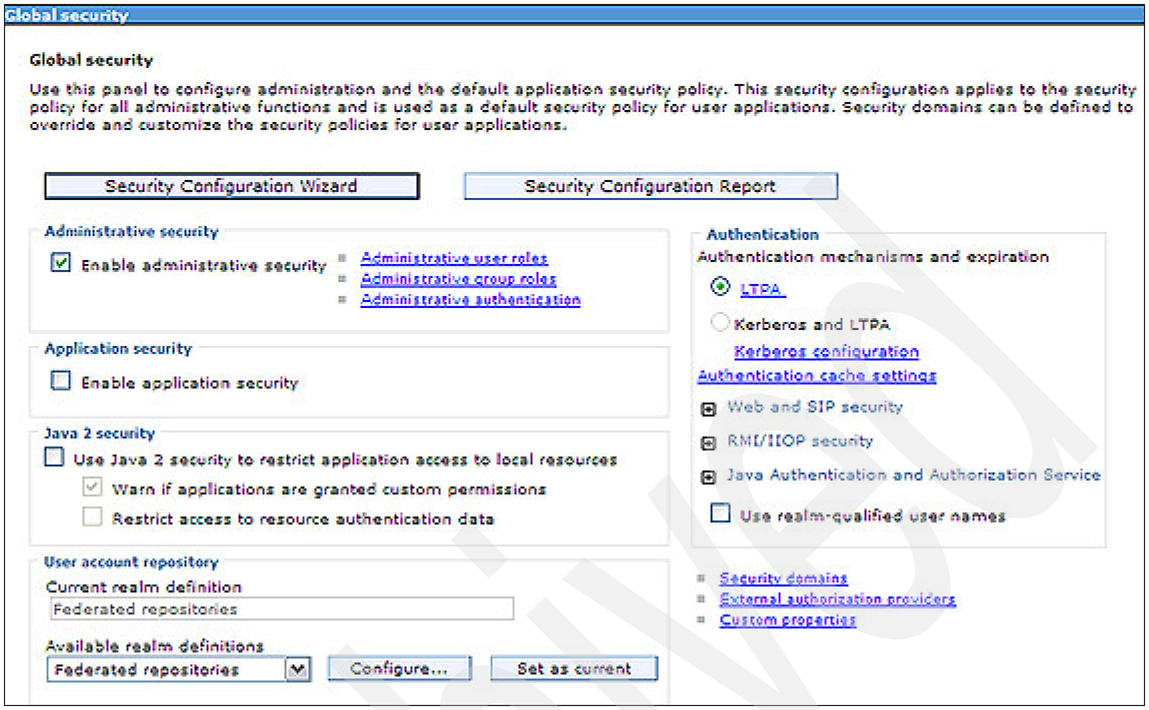
When you create a profile, you have the option to enable administrative security. This option is selected by default, and you are asked to provide a user ID and password for the administrator. This information is registered in a file-based repository. After the profile is created and the process hosting the administrative services is started, you must use this user ID and password to log in to the administrative console. Additional users and groups can be added to the file-based repository from the administrative console, and additional security configuration can be performed.

2.2.2 Enabling security after profile creation

If WebSphere administrative security has not been enabled, it can be configured and activated using the administrative console.

Follow these steps to enable administrative security:

1. Determine the user registry that you will use. If you plan to use a user registry other than the file-based (federated) repository, you must populate it with the user ID and password that you will specify as the admin user.
2. Select **Security** → **Global security** (as shown in Table 2-1 on page 28).



Global security

Global security

Use this panel to configure administration and the default application security policy. This security configuration applies to the security policy for all administrative functions and is used as a default security policy for user applications. Security domains can be defined to override and customize the security policies for user applications.

Security Configuration Wizard | Security Configuration Report

Administrative security

[✓] Enable administrative security

- Administrative user roles
- Administrative group roles
- Administrative authentication

Application security

[ ] Enable application security

Java 2 security

[ ] Use Java 2 security to restrict application access to local resources

[✓] Warn if applications are granted custom permissions

[ ] Restrict access to resource authentication data

User account repository

Current realm definition: Federated repositories

Available realm definitions: Federated repositories | Configure... | Set as current

Authentication

Authentication mechanisms and expiration

(●) LTPA

( ) Kerberos and LTPA

Kerberos configuration

Authentication cache settings

⊞ Web and SIP security

⊞ RMI/IIOP security

⊞ Java Authentication and Authorization Service

[ ] Use realm-qualified user names

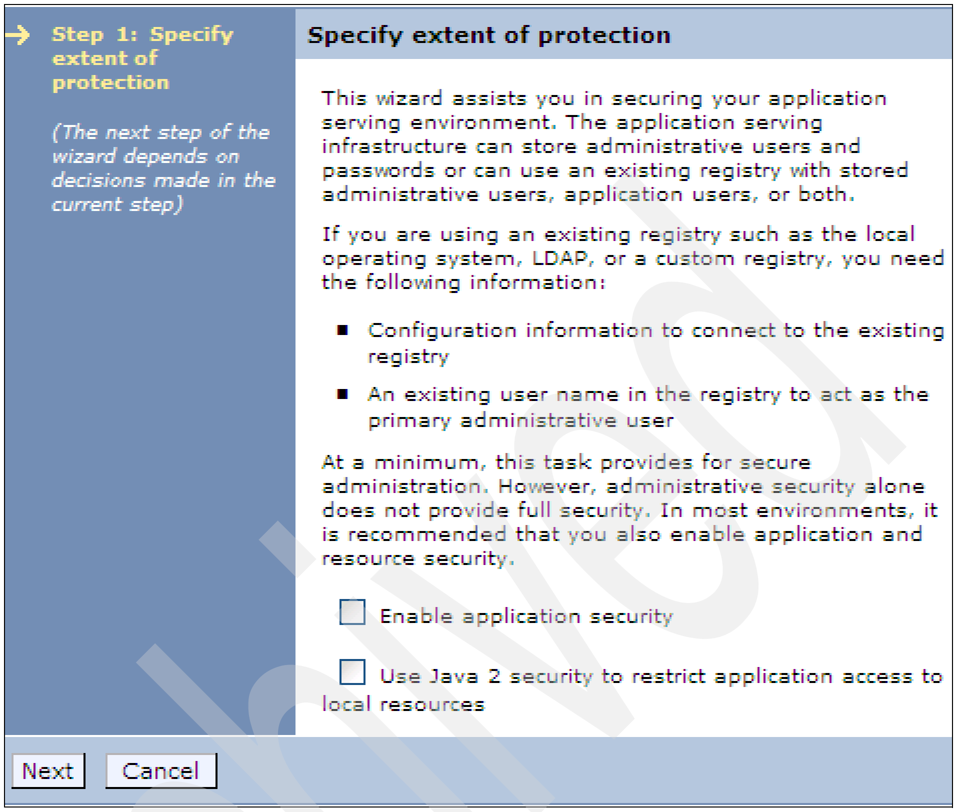
- Security domains
- External authorization providers
- Custom properties

Figure 2-1 Global security configuration page

3. The Security Configuration Wizard guides you through the process of completing the basic requirements to secure your application serving environment.

   Launch the wizard by clicking **Security Configuration Wizard**.

4. The first step of the wizard allows you the option of enabling application security and Java 2 security, in addition to administrative security (Figure 2-2 on page 21).


Step 1: Specify extent of protection

(The next step of the wizard depends on decisions made in the current step)

Specify extent of protection

This wizard assists you in securing your application serving environment. The application serving infrastructure can store administrative users and passwords or can use an existing registry with stored administrative users, application users, or both.

If you are using an existing registry such as the local operating system, LDAP, or a custom registry, you need the following information:

- Configuration information to connect to the existing registry
- An existing user name in the registry to act as the primary administrative user

At a minimum, this task provides for secure administration. However, administrative security alone does not provide full security. In most environments, it is recommended that you also enable application and resource security.

☐ Enable application security

☐ Use Java 2 security to restrict application access to local resources

Figure 2-2 Step 1 of Configuration Wizard

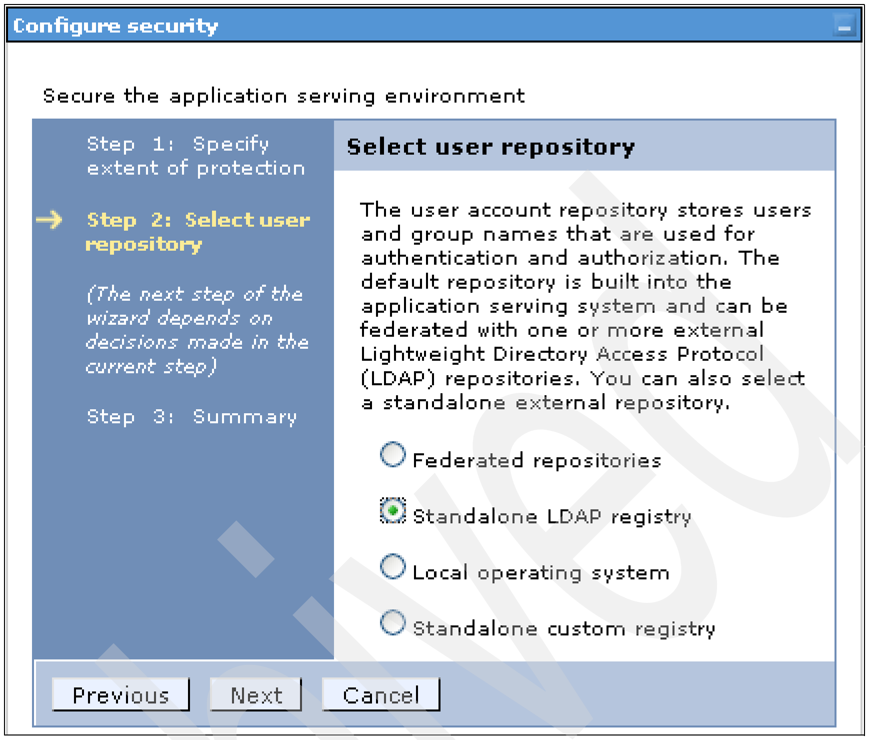
In this example, only administrative security will be enabled.

Application security is also essential in securing a WebSphere environment and is discussed in Chapter 7, “Application security” on page 253.

Java 2 security provides a policy-based, fine-grained access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources. Unless you have planned and prepared for using Java 2 in advance, many existing or even new applications might not be prepared for the extremely fine-grained access control programming model that Java 2 security is capable of enforcing. In addition, there is a permanent cost to Java 2 security. (It is not common to enable Java 2 security).

Click **Next**.

5. The second step of the wizard allows you to select the user repository (Figure 2-3 on page 22).


Configure security

Secure the application serving environment

Step 1: Specify extent of protection

→ Step 2: Select user repository

(The next step of the wizard depends on decisions made in the current step)

Step 3: Summary

Select user repository

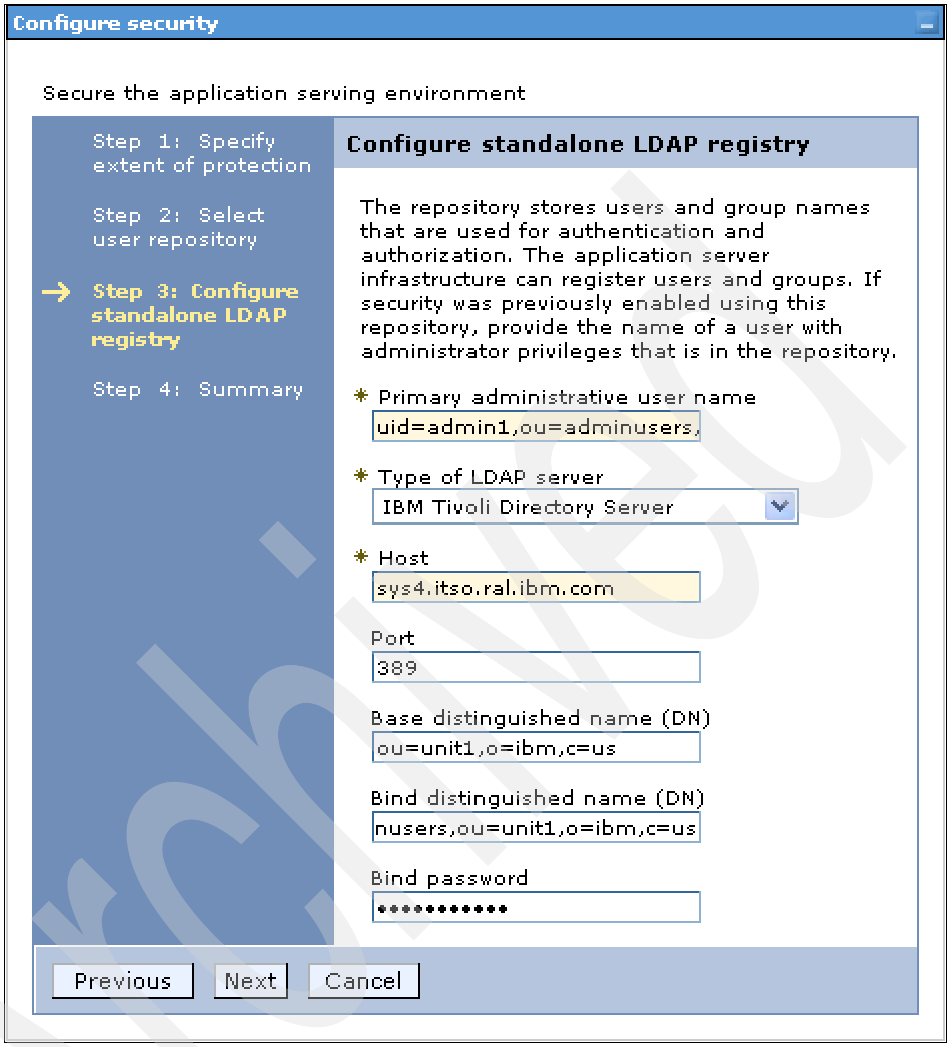
The user account repository stores users and group names that are used for authentication and authorization. The default repository is built into the application serving system and can be federated with one or more external Lightweight Directory Access Protocol (LDAP) repositories. You can also select a standalone external repository.

- ○ Federated repositories
- ● Standalone LDAP registry
- ○ Local operating system
- ○ Standalone custom registry

Figure 2-3 Select the repository

Click **Next**.

6. The third step of the configuration wizard allows you to provide a primary administrative user and other user registry information (Figure 2-4 on page 23).


Configure security

Secure the application serving environment

Step 1: Specify extent of protection
Step 2: Select user repository
Step 3: Configure standalone LDAP registry
Step 4: Summary

Configure standalone LDAP registry

The repository stores users and group names that are used for authentication and authorization. The application server infrastructure can register users and groups. If security was previously enabled using this repository, provide the name of a user with administrator privileges that is in the repository.

* Primary administrative user name
uid=admin1,ou=adminusers,

* Type of LDAP server
IBM Tivoli Directory Server

* Host
sys4.itso.ral.ibm.com

Port
389

Base distinguished name (DN)
ou=unit1,o=ibm,c=us

Bind distinguished name (DN)
nusers,ou=unit1,o=ibm,c=us

Bind password
●●●●●●●●●●●

Figure 2-4 Enter the primary administrative user

You need to enter a valid user name in the Primary administrative user name field. The primary administrative user is a member of the chosen repository, but it also has the same privileges that are associated with the administrative role ID in WebSphere Application Server, and it can access all of the protected administrative methods.

Figure 2-4 shows the single LDAP configuration. Based on the registry you choose in the previous step, other information in addition to the primary administrative user will be required to configure the registry.

If local operating system registry is selected, the ID must have the following platform-specific privileges:

- For Windows operating systems: Act as Part of Operating System privileges
- For UNIX and Linux® operating systems: Root privileges

Click **Next** to view the summary page, and then complete the wizard.

Apply the changes; administrative security will be turned on by default.

7. Validate the completed security configuration by clicking **OK** or **Apply**. If there are no validation problems, click **Save** to save the settings to a file that the server uses when it restarts.

Important:

- If you do not click **Apply** or **OK** in the Global security panel before you click **Save**, your changes are not written to the workspace.
- The server must be restarted for any changes to take effect when you start the administrative console.

Verifying and testing administrative security

After your server has been restarted in secure mode, you can test that security is properly enabled. There are several basic tests that you can perform:

- Verify the form login. When using the administrative console, the login page that is displayed forces you to fill in a user ID and a password. Only a user ID with one of the administrative roles can be able to log in.
- Verify that the Java Client Basic Authentication works by executing:

  `<WebSphere_home>\bin\dumpNameSpace.bat`

  The connection testing the dumpNameSpace script needs to be configured to spawn a window (for example, export DISPLAY or something else appropriate based on the operating system).

  A challenge login window must open. Although you might be able to just click **Cancel**, you must type *any* correct user ID and password defined in the user account repository with the administrator role to test the security.

  Be aware that the login panel for the previously mentioned Java client only opens if the property `com.ibm.CORBA.loginSource` is set to `prompt` in the file `sas.client.props`. Clicking **Cancel** works only if the CosNaming security (refer to 2.7, "Naming service security: CosNaming roles" on page 46) allows *read access* to everyone. These values are the default values when you installed WebSphere.

Successfully running the previously listed basic tests indicates that the administrative security is working correctly.

2.2.3 Stopping the application server

While the command to start the application server is still the same when administrative security is enabled, stopping the server requires extra information. You have to specify a user ID with administrator role rights, or you have to specify the primary administrative user name specified in the user account repository and its password.

The most secure way to stop the application server is to enter the **stopServer** command without the user ID and password, and enter them at the command prompt:

```
install_root\bin\stopServer.bat <server_name>
```

Depending on your requirements and environment, there are alternate methods to enter the user ID and password.

You can them in the **stopServer** command:

```
install_root\bin\stopServer.bat <server_name> -username <user ID>
-password <password>
```

For WebSphere Application Server running under a UNIX®-based operating system (OS), the previously mentioned command (the UNIX equivalent) carries a serious security problem. Anybody who uses the command **ps -ef** while the `stopServer` process is running is able to see the user ID and the password.

To avoid this problem:

1. If you are using the SOAP connection type (default) to stop the server, edit the following file:

   ```
   profile_home\properties\soap.client.props
   ```

   Then, change the values of these properties:

   ```
   com.ibm.SOAP.securityEnabled=true
   com.ibm.SOAP.loginUserid=<user ID>
   com.ibm.SOAP.loginPassword=<password>
   ```

   Again, the user ID <*user ID*>, with its password <*password*>, is the user ID with administrator role rights or the primary administrative user name defined in the user account repository.

2. We recommend that you encode the `com.ibm.SOAP.loginPassword` property value using:

   ```
   <WebSphere_home>\bin\PropFilePasswordEncoder.bat soap.client.props
   com.ibm.SOAP.loginPassword
   ```

   Examine the result and remove the backup file, `soap.client.props.bak`, which was created by the command that was used previously. It contains the unencrypted password.
3. Make sure that proper file access rights for sensitive WebSphere Application Server files, such as properties files and executable files, are set. At a minimum, ensure that the permissions prevent general users from accessing these files. WebSphere administrators must be the only users that are granted access to these files. For optimal security, access to the entire WebSphere directory tree must be removed for general users.

2.3 Disabling administrative security

Note: Disabling administrative security does not disable Java 2 security automatically. Java 2 security needs to be disabled explicitly.

Administrative security can be disabled in order to fix a problem that stems from a situation in which WebSphere security is failing.

If you can log on to the administrative console, disabling security is fairly easy. Follow these steps:

1. Click **Security** → **Global security**.
2. Clear the **Enable administrative security** check box.
3. Save the configuration and restart the server.

If, for some reason, the server hosting the administrative console cannot be started, for example, because of an incorrectly configured user account repository, you can disable administrative security using the command line:

1. At the command prompt, type the following:

   ```
   <WebSphere_home>\bin\wsadmin.bat -conntype NONE
   ```

2. When the system command prompt displays again, type:

   ```
   securityoff
   ```

3. When done, type `exit` and restart the application server.

This procedure typically works without any problem, but in the event that it fails, you can disable administrative security by directly editing the `security.xml` file (Example 2-1) in the following location:

`*profile_home*\config\cells\<*cell_name*>\`

Open this file and change the security attribute `enabled=true` to `enabled=false`.

Other security properties, such as Java 2 security and application security, can also be found in this file. Only modify the XML file as a last resort. Be sure to save an original copy of the `security.xml` file before making any modifications.

Example 2-1 Content snippet of the file security.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<security:Security xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
...
xmi:id="Security_1" useLocalSecurityServer="true"
useDomainQualifiedUserNames="false" enabled="true" cacheTimeout="600"
issuePermissionWarning="true" activeProtocol="BOTH"
enforceJava2Security="false" enforceFineGrainedJCASecurity="false"
appEnabled="false" dynamicallyUpdateSSLConfig="true"
allowBasicAuth="true" activeAuthMechanism="LTPA_1"
activeUserRegistry="WIMUserRegistry_1"
defaultSSLSettings="SSLConfig_1">
<authMechanisms ...
...
...
</security:Security>
```

2.4 Administrative roles

WebSphere Application Server extends the Java security role-based access control to protect the product administrative and naming subsystems. Eight administrative roles are available to define the administrative capabilities assigned to a user or group (Table 2-1 on page 28). The administrative roles are effective only when administrative security is enabled.

New in V7: The Auditor role has been introduced. Having this role allows an administrator to view and modify configuration settings related to the security auditing subsystem. This new role allows for the separation of administrative privileges.

Table 2-1 WebSphere administrative roles

Role	Description
Monitor	Least privileged. This role allows a user to view the WebSphere configuration and the current state of the application server.
Configurator	Monitor privilege plus the ability to change the WebSphere configuration.
Operator	Monitor privileges plus the ability to change the runtime state, such as starting or stopping servers.
Administrator	Operator, configurator, and iscadmins privilege, plus privileges granted solely to the administrator role, such as: ► Modifying the primary administrative user and password ► Creating, updating, and deleting users and groups ► Enabling or disabling administrative and Java 2 security **Note:** An administrator cannot map users and groups to administrative roles.
iscadmins	*Only available for administrative console users and not for wsadmin users.* Allows a user to manage users and groups in the federated repositories.
Deployer	*Only available for wsadmin users and not for administrative console users.* Allows a user to change the configuration and the runtime state on applications using **wsadmin**.
Admin Security Manager	Allows a user to map users and groups to administrative roles through the administrative console, or through **wsadmin** for fine-grained security. When fine-grained administrative security is used, users granted this role can manage authorization groups.

Role	Description
Auditor	Users granted this role can view and modify the configuration settings for the security auditing subsystem. For example, a user with the auditor role can complete the following tasks: ► Enable and disable the security auditing subsystem. ► Select the event factory implementation to be used with the event factory plug-in point. ► Select and configure the service provider, emitter, or both to be used with the service provider plug-in point. ► Set the audit policy that describes the behavior of the application server in the event of an error with the security auditing subsystem. ► Define which security events are to be audited. The auditor role includes the monitor role, which allows the auditor to view but not change the rest of the security configuration.

Important information: The primary administrative user that is specified when enabling administrative security is automatically mapped to the administrator and AdminSecurityManager roles. Therefore, it is not necessary to manually add this identity to either of these administrative roles.

The primary administrative user is also given the auditor role initially. Users who want to separate the administrative privileges between audit configuration and the remaining security configuration must define a user other than the primary administrative user as their primary auditor user.

2.4.1 Mapping users and groups to administrative roles

Users and groups can be added or removed from administrative roles using the WebSphere Application Server administrative console by a user given the appropriate authority. The Primary administrative user name must be used to log on to the administrative console to change the administrative user and group roles other than the auditor role. Only a user with the auditor role can change the auditor user and group roles. When security auditing is initially enabled, the Primary administrative user is also given the auditor role, and can manage all of the administrative user and group roles, including those users in the auditor role.

Best practice: Map a group or groups, rather than specific users, to administrative roles, because it is more flexible and easier to administer. When users are added to the groups (therefore, the users are mapped to administrative roles), it will be effective without the requirement to restart the WebSphere server.

In addition to mapping users or groups, a special subject can also be mapped to the administrative roles. A *special subject* is a generalization of a particular class of users. The AllAuthenticated special subject means that the access check of the administrative role ensures that the user making the request is at least authenticated. The Everyone special subject means that anyone, authenticated or not, can perform the action, as though security was not enabled.

A suggested process for assigning administrative security access to users is:

1. Create groups that correspond to the administrative roles.

   This step is done using the appropriate interface to the user registry.

   If you are using the federated repository, you can add groups using the panels found by selecting **Users and Groups** → **Manage Groups** in the administrative console.

2. Create user IDs for the administrative users and add each user ID to the appropriate group.

   This step is also done using the appropriate interface to the user registry.

   If you are using the federated repository, you can add users and assign them to groups using the panels found by selecting **Users and Groups** → **Manage Users** in the administrative console.

3. Assign administrative roles to the groups.

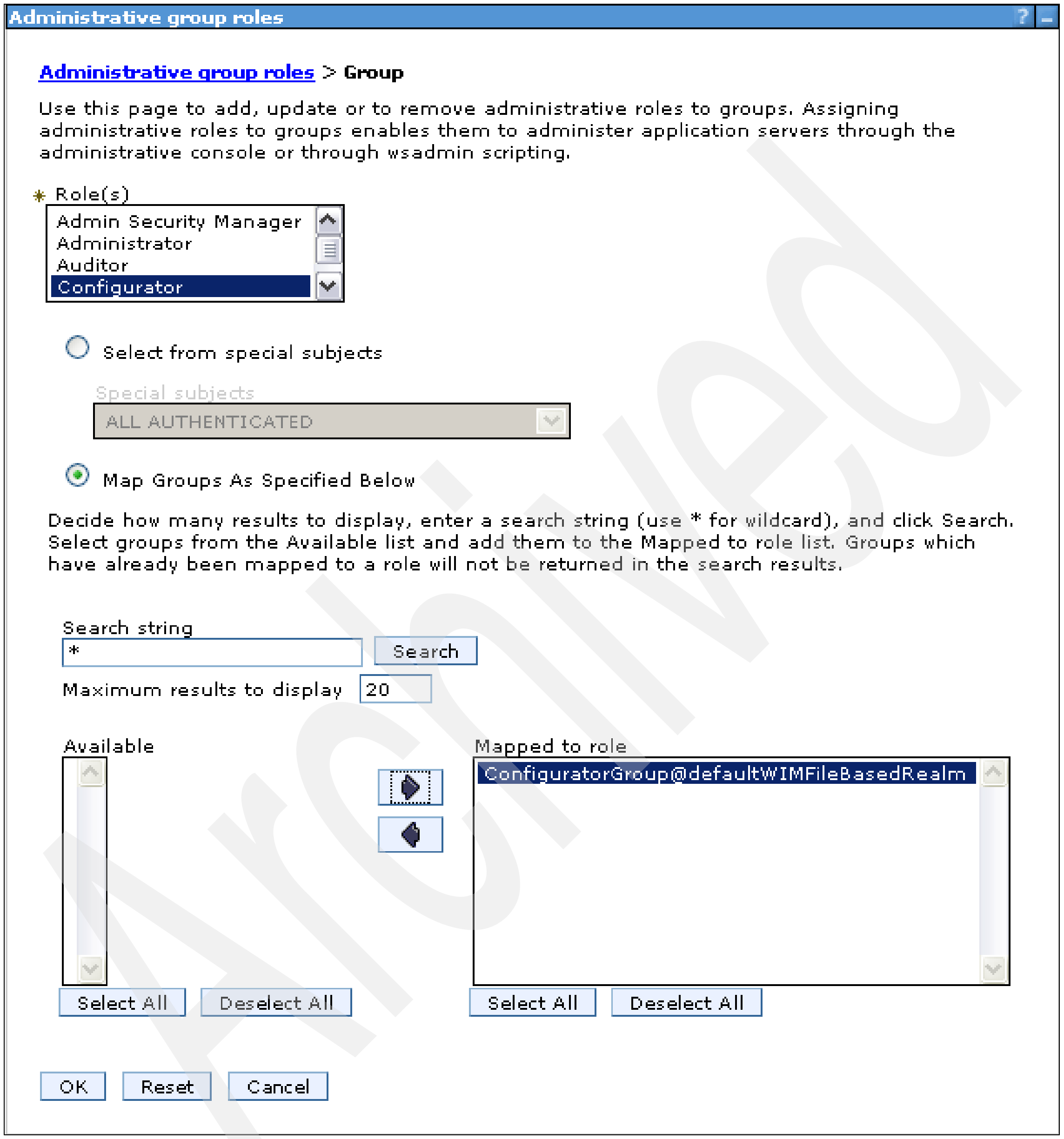
   The panels that allow you to assign administrative roles to groups can be found in the administrative console by selecting **Users and Groups** → **Administrative group roles**.

2.4.2 Mapping a group to an administrative role

You can use the following process to map an existing group to an administrative role:

1. From the administrative console, click **Users and Groups** → **Administrative Group Roles**.
2. Click **Add**.
3. Either a specific group, or a special subject can be mapped. Refer to Figure 2-5 on page 32.

   To map a specific group, select **Map Groups As Specified Below**, follow the instructions on the page to specify a group, and then select the appropriate administrative role. More than one role can be selected by using the Ctrl key.


Administrative group roles

Administrative group roles > Group

Use this page to add, update or to remove administrative roles to groups. Assigning administrative roles to groups enables them to administer application servers through the administrative console or through wsadmin scripting.

* Role(s)

Admin Security Manager
Administrator
Auditor
Configurator

( ) Select from special subjects

Special subjects
ALL AUTHENTICATED

(●) Map Groups As Specified Below

Decide how many results to display, enter a search string (use * for wildcard), and click Search. Select groups from the Available list and add them to the Mapped to role list. Groups which have already been mapped to a role will not be returned in the search results.

Search string: *

Maximum results to display: 20

Available

Mapped to role: ConfiguratorGroup@defaultWIMFileBasedRealm

Figure 2-5 Mapping a group to an administrative role

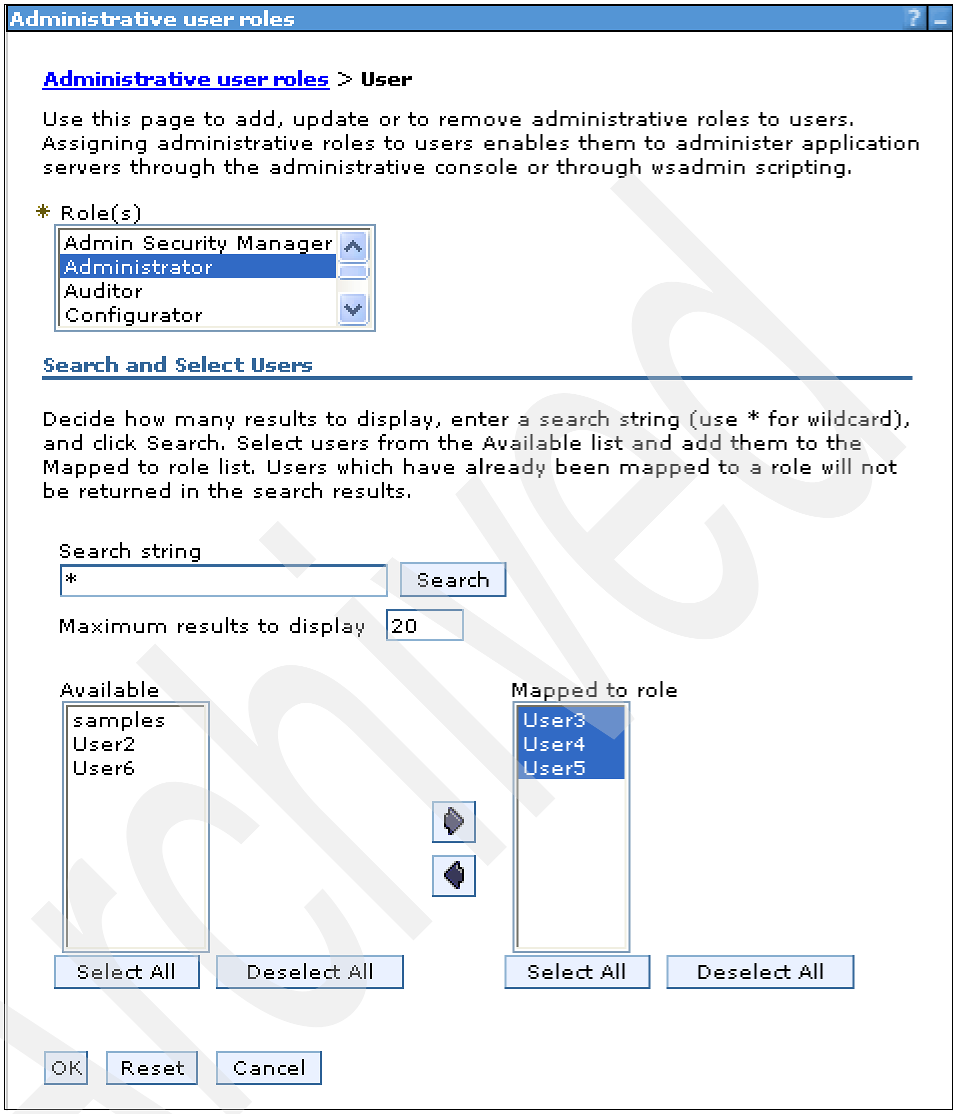
To map a special subject, select **Special subjects** and the appropriate subject from the drop-down list. A *special subject* is a generalization of a particular class of users. The AllAuthenticated special subject means that the access check of the administrative role ensures that the user making the request is at least authenticated. The Everyone special subject means that anyone, authenticated or not, can perform the action, as though security was not enabled. The AllAuthenticatedInTrustedRealms special subject is similar to the AllAuthenticated special subject that is currently supported. The difference is that the AllAuthenticated special subject refers to users in the same realm as the application while the AllAuthenticatedInTrustedRealms special subject applies to all of the users in the trusted realms and in the realm of the application.

4. Click **OK**.
5. Ensure that the new mapping is in the Administrative Group Roles list, and then click **Save** to save the change to the master configuration.

2.4.3 Mapping a user to an administrative role

In order for a user to perform an administrative action, the user's identity must be mapped to an administrative role:

1. From the administrative console, select **Users and Groups** → **Administrative User Roles**.
2. Click **Add** to add a user.
3. Select the appropriate administrative role. More than one role can be selected by using the Ctrl key. Refer to Figure 2-6 on page 34. To select the roles:
   a. Enter the search string, use an asterisk (*) for a wildcard, and click **Search**.
   b. Select the users from the **Available** user list. You can use the Ctrl or Shift key to select multiple users.
   c. Click the add arrow button to add the users to the **Mapped to role** user list.
   d. Select the users from the **Mapped to role** user list. You can use the Ctrl or Shift key to select multiple users, and then select the role from the **Role(s)** list.



Use this page to add, update or to remove administrative roles to users. Assigning administrative roles to users enables them to administer application servers through the administrative console or through wsadmin scripting.

Decide how many results to display, enter a search string (use * for wildcard), and click Search. Select users from the Available list and add them to the Mapped to role list. Users which have already been mapped to a role will not be returned in the search results.

Search string: *

Maximum results to display: 20

Figure 2-6 Mapping a user to an administrative role

4. Click **OK**.
5. Ensure that the new mapping is in the Administrative user roles list, and then click **Save** to save the change to the master configuration.
6. You might need to restart the server for the role to become available to the user, especially if the user has been active. Restarting the server ensures that the cache is refreshed with the new role information.

2.5 Fine-grained administrative security

In releases prior to WebSphere Application Server Version 6.1, users who were granted administrative roles administered all of the resource instances in the cell. WebSphere Application Server is now more fine-grained, meaning that administrative roles can now be assigned per resource instance rather than to the entire cell. Users cannot perform actions on resources outside of those resources that are assigned to them. Users can be granted configurator access to a specific instance of a resource (an application, an application server, or a node).

There is a cell-wide authorization group for backward compatibility. Users assigned to administrative roles in the cell-wide authorization group can still access all of the resources within the cell. When using fine-grained security, you can assign users the cell-wide monitor role so that they can see resources without being able to update those resources outside of their fine-grained authorization role.

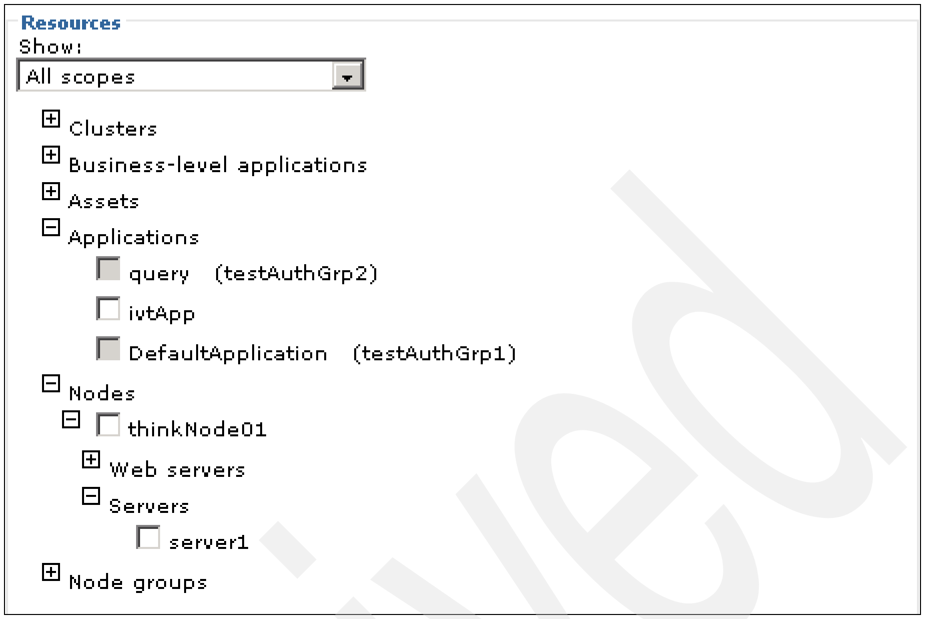
2.5.1 Authorization group

To achieve the instance-based security or fine-grained security, resources that require the same privileges are placed in a group called the *administrative authorization group* or *authorization group*. Users can be granted access to the authorization group by assigning the required administrative role to them.

The following types are valid for the resource instances that are added to an authorization group:

- Cluster
- Node
- Servers, including application servers and Web servers
- Applications, including business-level applications
- Node groups
- Assets

A resource instance can only belong to one authorization group. When you create a new authorization group through the administrative console, you can list the resource in different scopes, and if one resource has been added to an authorization group, it cannot be selected for another authorization group, as shown in Figure 2-7 on page 36.


Resources

Show:

All scopes

- Clusters
- Business-level applications
- Assets
- Applications
  - query (testAuthGrp2)
  - ivtApp
  - DefaultApplication (testAuthGrp1)
- Nodes
  - thinkNode01
    - Web servers
    - Servers
      - server1
- Node groups

Figure 2-7 List resources with different scopes

However, there is a containment relationship among resource instances. If a parent resource belongs to a different authorization group than that of its child resource instance, the child resource instance implicitly belongs to multiple authorization groups.

For example, when you assign a node to an authorization group, it includes all of the resources associated with this node. Then, if you assign the child resource server to another authorization group, this server resource will belong to multiple authorization groups.

2.5.2 Granting fine-grained access

There are two steps in granting fine-grained administration authority to users:

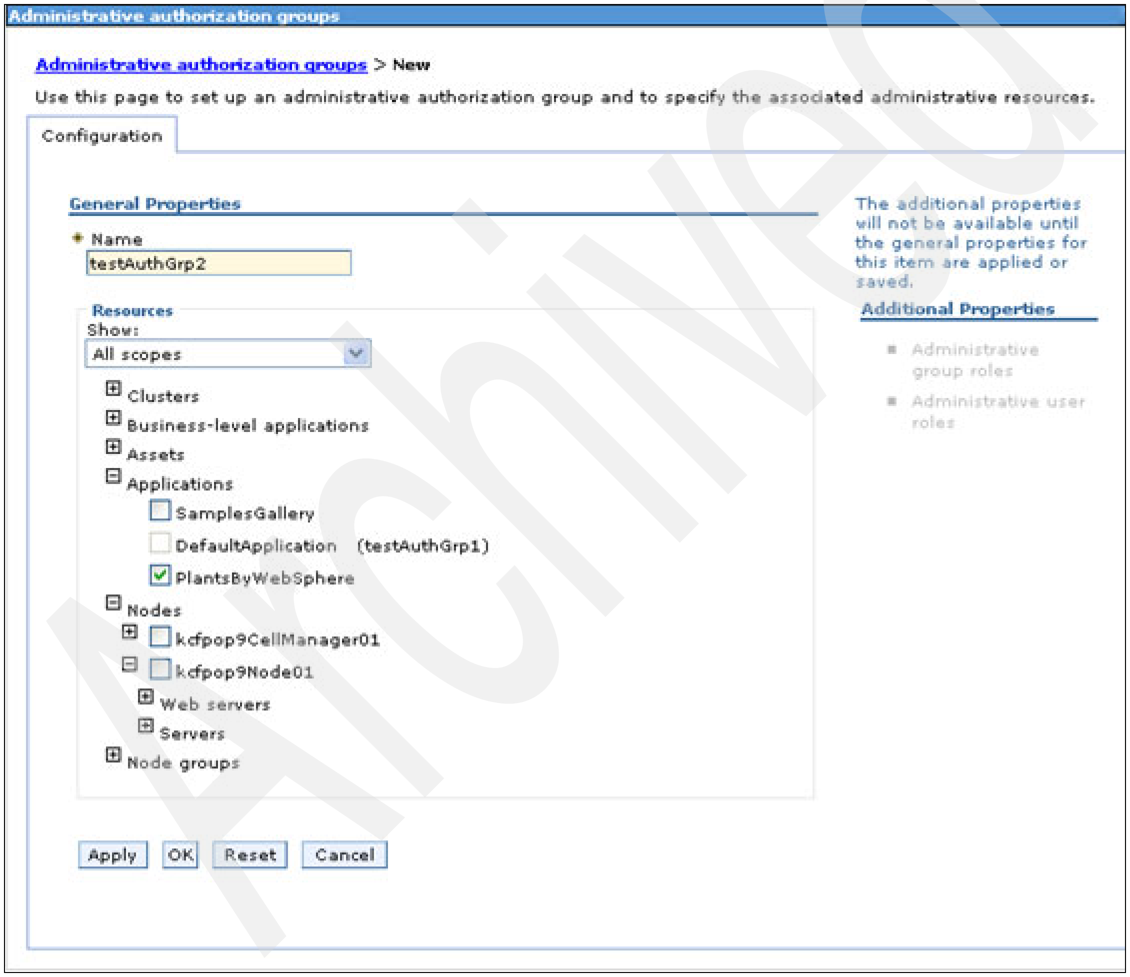
1. Create an authorization group and map resources to it.
2. Assign users or groups to administrative roles within the authorization groups, using **`wsadmin`** or the administrative console.

You can find examples of **`wsadmin`** commands that can be used to manage fine-grained access at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_fineg_admsec.html`

The following steps show how to configure fine-grained security through the administrative console:

1. Log in to the administrative console, and click **Security** → **Administrative Authorization Groups**.
2. Click **New** to create a new administrative authorization group.
3. Type the group name and select the resources. Refer to Figure 2-8. You can use the Show: field to narrow the resource type that is displayed. For example, select Servers to show all the servers in the cell.


Administrative authorization groups

Administrative authorization groups > New

Use this page to set up an administrative authorization group and to specify the associated administrative resources.

Configuration

General Properties

* Name: testAuthGrp2

Resources

Show: All scopes

- ⊞ Clusters
- ⊞ Business-level applications
- ⊞ Assets
- ⊟ Applications
  - ☐ SamplesGallery
  - ☐ DefaultApplication (testAuthGrp1)
  - ☑ PlantsByWebSphere
- ⊟ Nodes
  - ⊞ ☐ kcfpop9CellManager01
  - ⊟ ☐ kcfpop9Node01
    - ⊞ Web servers
    - ⊞ Servers
- ⊞ Node groups

The additional properties will not be available until the general properties for this item are applied or saved.

Additional Properties

- Administrative group roles
- Administrative user roles

Figure 2-8 Administration authorization groups

4. Click **Apply**.

5. Now, you can click **Administrative user roles** or **Administrative group roles** to assign users or groups to administrative roles within this authorization group. Refer to 2.4.3, "Mapping a user to an administrative role" on page 33 and 2.4.2, "Mapping a group to an administrative role" on page 31 to assign users or groups to administrative roles. Note that the users and groups that you assign must also have the monitor administrative role if they will use the administrative console.
6. Click **Save** to save the change to the master configuration.

2.5.3 Using fine-grained security: An example

This section shows a simple, practical example of the use of fine-grained security for the deployer role.

In this example, there are three applications (App1, App2, and App3) deployed on server1. Each application must be isolated so that the administrator of one application cannot modify another application. Assume that only user1 can manage application App1, that only user2 can manage application App2, and that App3 only can be managed by the cell-level administrator, as shown in Figure 2-9.

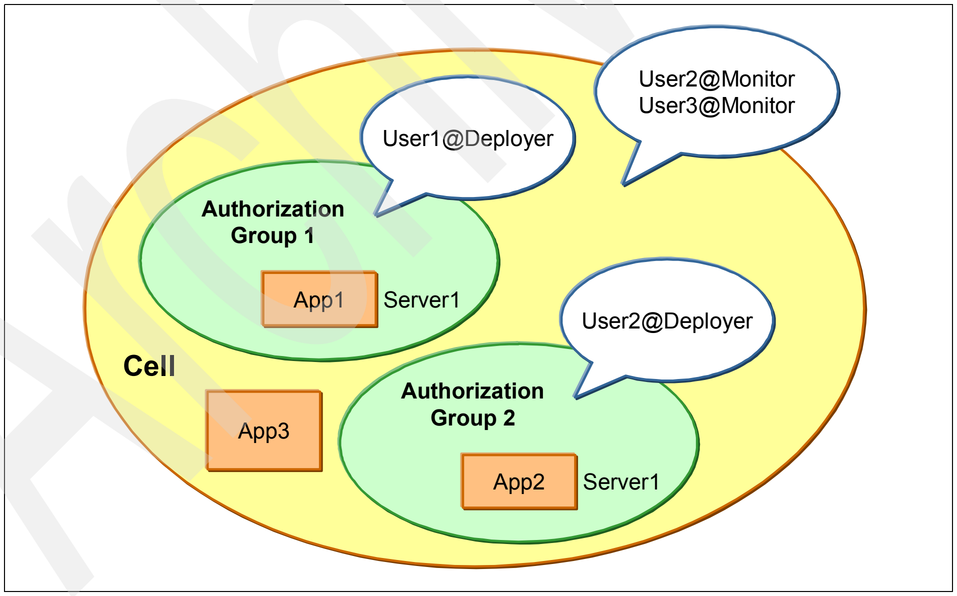


Figure 2-9 Fine-grained security scenario

These steps illustrate how fine-grained security works using the scenario in Figure 2-9 on page 38:

- Two authorization groups are created. Application App1 is in Group1, and application App2 is in Group2.
- Application App3 does not belong to any authorization group.
- A deployer role is assigned from authorization group Group1 to user1 and also from authorization group Group2 to user2.
- A monitor role is assigned to user2 and user3 at the cell level.

Security levels: To use fine-grained administrative security in the administrative console, a user must be granted the monitor role at the cell level at minimum. If the administrator gives users access only to specific authorization groups or permissions to non-cell authorization groups, the users must use wsadmin.

Consequently, user1 can perform all of the operations on application App1, user2 on application App2, and user2 and user3 have the monitor role for the cell level.

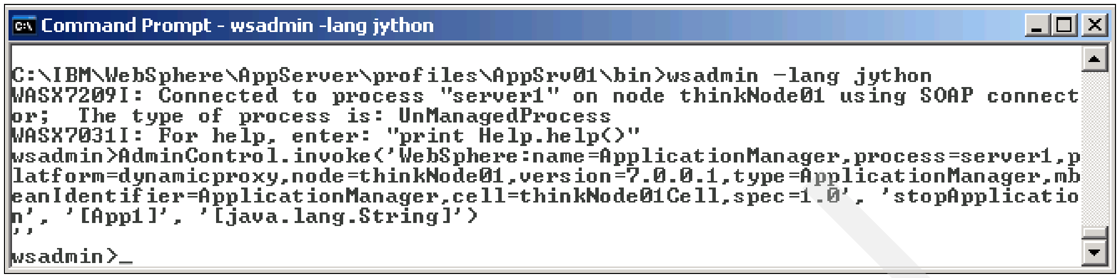
Because all applications share the same server, the same server cannot be put in all authorization groups. Only a cell-level administrator can install an application. After the installation of an application is complete, the deployer of each application can modify their own application. To start and stop the server, cell-level administrative authority is required.

When user1 tries to log in to the administrative console, because there is no role assigned at the cell level, no tasks can be performed. The message that is shown in Example 2-2 will be displayed.

Example 2-2 Message when user1 tries to log in to the administrative console

```
Cell wide monitor role is minimally required to effectively use the
administrative console. Assign cell level role to this user or group if
administrative console access is required.
```

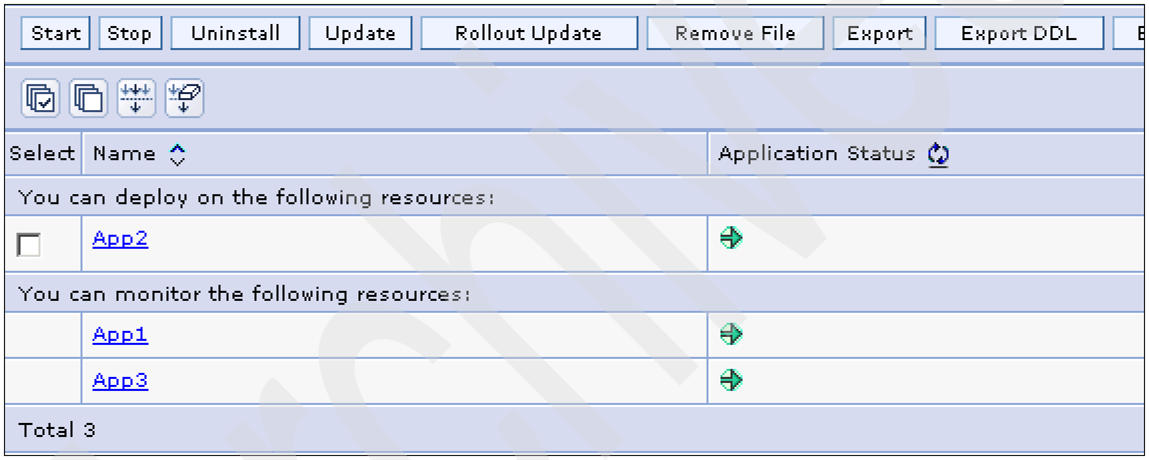
User1 must use wsadmin to perform the operations on application App1. Figure 2-10 on page 40 shows how user1 uses wsadmin to stop App1.



```
C:\IBM\WebSphere\AppServer\profiles\AppSrv01\bin>wsadmin -lang jython
WASX7209I: Connected to process "server1" on node thinkNode01 using SOAP connect
or;  The type of process is: UnManagedProcess
WASX7031I: For help, enter: "print Help.help()"
wsadmin>AdminControl.invoke('WebSphere:name=ApplicationManager,process=server1,p
latform=dynamicproxy,node=thinkNode01,version=7.0.0.1,type=ApplicationManager,mb
eanIdentifier=ApplicationManager,cell=thinkNode01Cell,spec=1.0', 'stopApplicatio
n', '[App1]', '[java.lang.String]')
''
wsadmin>_
```

Figure 2-10   Stop App1 using wsadmin

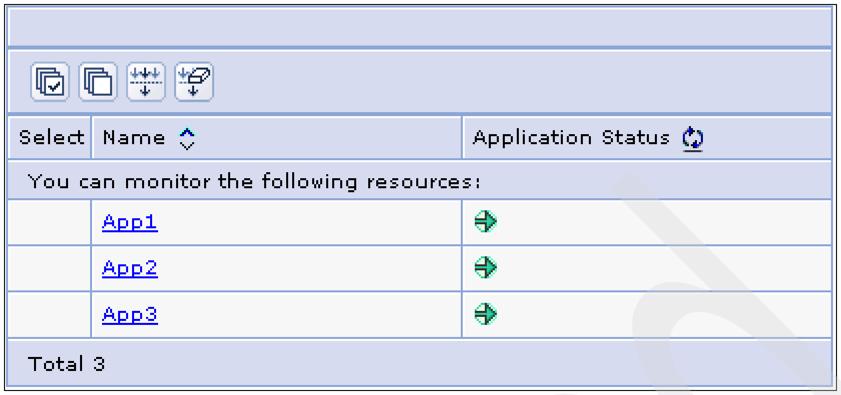
Because user2 has the cell-wide monitor role and the deployer role on application App2, this user can use the fine-grained security in the administrative console in addition to using wsadmin, as shown in Figure 2-11.



Select	Name	Application Status
You can deploy on the following resources:		
☐	App2	➡
You can monitor the following resources:		
	App1	➡
	App3	➡
Total 3		

Figure 2-11   Fine-grained security for App2

And, user3 only has the cell-wide monitor role and no other role within an authorization group; therefore, this user can only monitor the status as shown in Figure 2-12 on page 41.



Select	Name	Application Status
You can monitor the following resources:		
	App1	➜
	App2	➜
	App3	➜
Total 3		

Figure 2-12 Monitor status for user3

2.6 Job manager security

Flexible management features have been introduced in WebSphere V7. You can use the flexible management environment to locally or remotely submit and manage administrative jobs. A job manager allows you to submit administrative jobs asynchronously for application servers registered to administrative agents and for deployment managers. You can submit these jobs to a large number of servers over a geographically dispersed area.

Two new server types have been added to support flexible management: the administrative agent and the job manager. The job manager is absolutely central to flexible management. The administrative agent is used for a base server to participate in flexible management. To participate in flexible management, a base server first registers itself with the administrative agent and then registers with the job manager. The deployment manager registers directly with the job manager; no administrative agent is involved in this case.

The job manager is used to queue jobs to application servers. These queued jobs are pulled from the job manager by the administrative agent and distributed to the appropriate application server or servers as shown in Figure 2-13 on page 42.

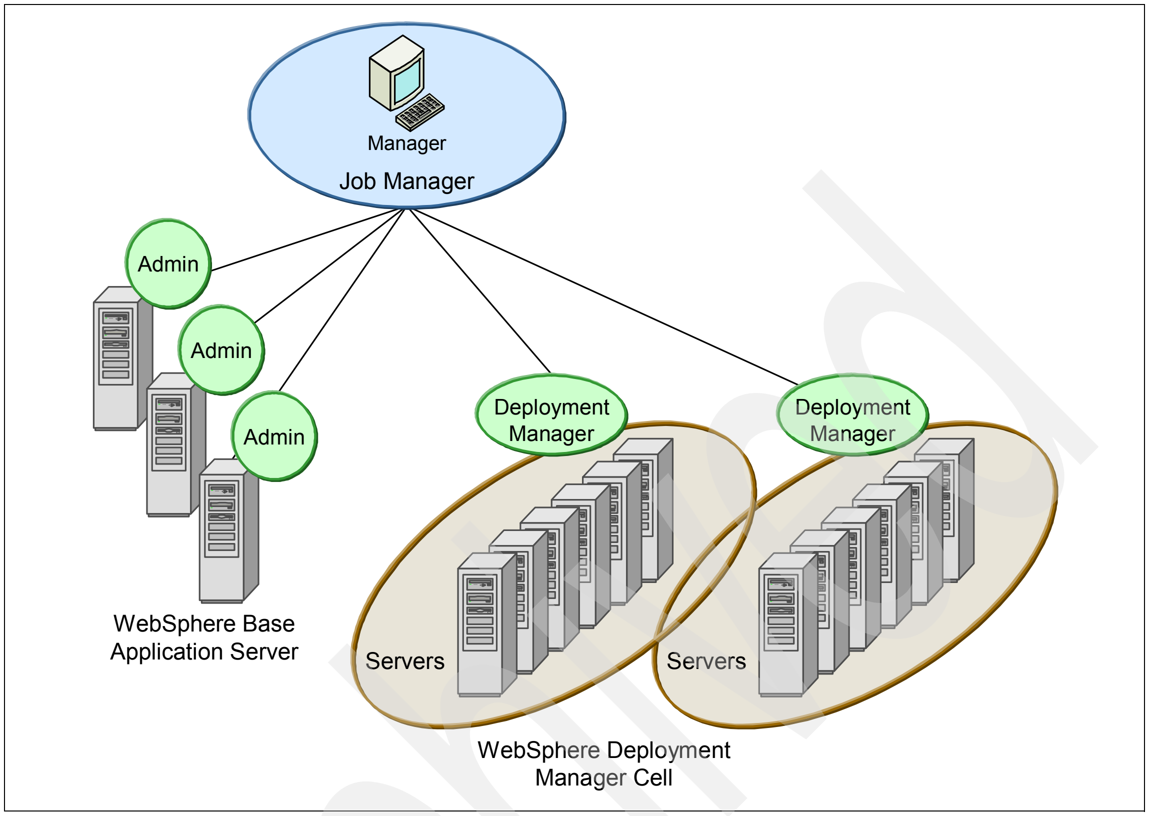


Figure 2-13 Job Manager

Enabling security in a flexible management environment is similar to the way that administrative security is enabled for servers and deployment managers. Security can be enabled during or after profile creation. However, in a flexible management environment, if security is to be enabled, it must be enabled before registration with the administrative agent or job manager.

Security for flexible management environments: In a flexible management environment, both the administrative agent and the base application server must have the same security level; that is, they must both have security enabled or disabled. We recommend that before registering a node with an administrative agent that you enable administrative security for both processes. After you register a profile with the administrative agent, the state of administrative security enablement cannot be changed.

When you access the WebSphere system in a Network Deployment environment, a single username and password is used to access all systems, but in a flexible management environment each job manager, administrative agent, and application server can have a separate username and password combination. If both the job manager and the managed node have security enabled, you will need to provide both sets of security credentials to submit a job to the application server in the managed node. To access the job manager's administrative interfaces, such as the administrative console, you need to provide the job manager's username and password. To submit the job to the application server, you need to provide the application server's username and password.

Figure 2-14 on page 44 shows the Job Manager administrative console and the job types that you can submit for processing.

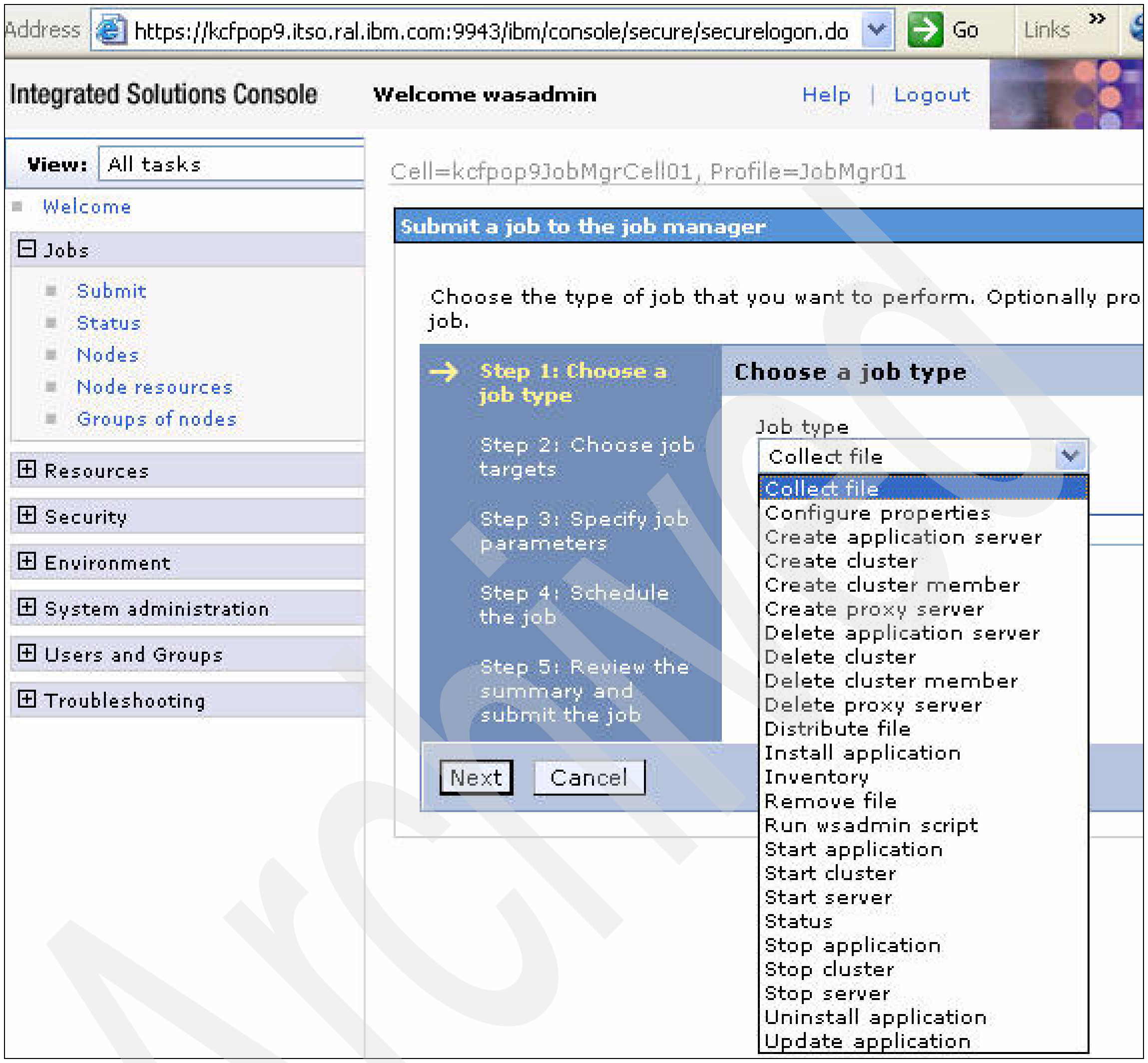


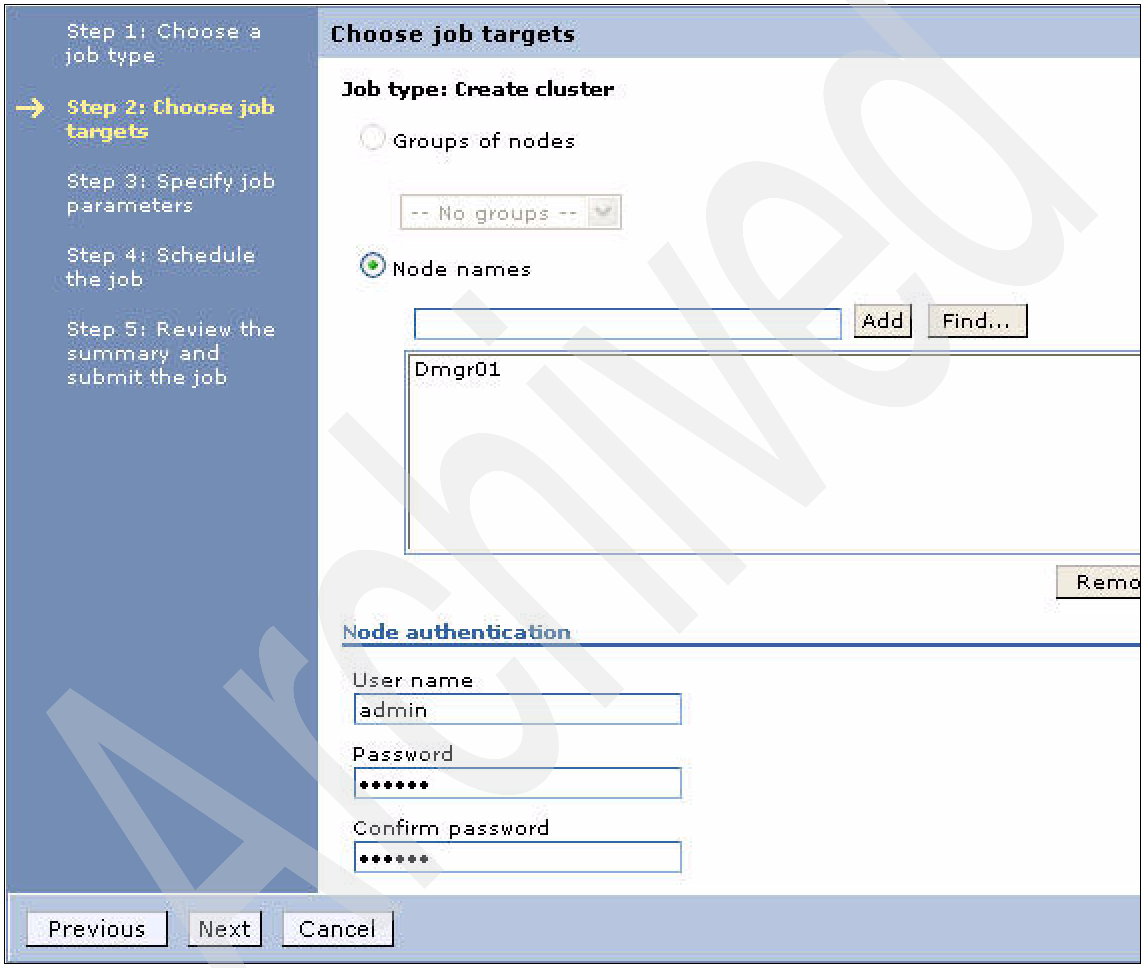
Figure 2-14 Job Manager Administrative Console

The required administrative roles for executing flexible management jobs are defined by the underlying administrative commands that are used by those jobs. For example, the required role for starting and stopping servers is the operator role. The operator role is also required for the execution of the flexible management jobs that start and stop servers. The general rules for assigning required administrative roles are:

- Viewing data requires the monitor role.
- Updating data requires the configurator role.

- Managing jobs requires the operator role.
- Registering or unregistering managed nodes requires the administrator role.

But this role management is only for the Job Manager. When you submit the job to the application server, you need to provide the application server's username and password combination that has the required authority to perform the requested operation (Figure 2-15).



Step 1: Choose a job type
Step 2: Choose job targets
Step 3: Specify job parameters
Step 4: Schedule the job
Step 5: Review the summary and submit the job
Choose job targets
Add
Find...
Remo
Previous
Next
Cancel

Job type: Create cluster

Groups of nodes

-- No groups --

[✓ Node names]

Dmgr01

Node authentication

User name: admin

Password: ••••••

Confirm password: ••••••

Figure 2-15 Submit a Job

If the user name specified does not have the required privileges, the job fails to be processed and a message similar to the message in Example 2-3 on page 46 occurs.

Example 2-3 Incorrect privileges message

```
CWWSY0605E: Problem while executing job createCluster. Cause:
java.lang.SecurityException: ADMF0010E: Access denied for command
createCluster.
```

2.7 Naming service security: CosNaming roles

The JEE role-based authorization concept has been extended to protect the WebSphere Common Object Request Broker Architecture (CORBA) naming service (CosNaming) to increase the granularity of its security control. In doing so, WebSphere is able to provide better control for a client program accessing the content of the WebSphere name space. There are generally two ways in which client programs make a CosNaming call:

- Through the Java Naming and Directory Interface (JNDI)
- CORBA clients invoking CosNaming methods directly

The default setup for WebSphere grants a Cos Naming Read role to the CosNaming service for everyone, which is the default setup for WebSphere. Table 2-2 shows all the four CosNaming roles.

Table 2-2 CosNaming roles

Role	Description
Cos Naming Read	Users are allowed to perform queries of the WebSphere name space, such as through the JNDI lookup method. The special subject Everyone is the default policy for this role.
Cos Naming Write	Users are allowed to perform write operations, such as JNDI bind, rebind, or unbind, and also CosNamingRead operations. The special subject, AllAuthenticated, is the default policy for this role.
Cos Naming Create	Users are allowed to create new objects in the name space through operations, such as JNDI create subcontext, and to perform CosNamingWrite operations. The special subject AllAuthenticated is the default policy for this role.
Cos Naming Delete	Users are able to destroy objects in the name space, for example, using the JNDI destroySubcontext method, as well as to perform CosNamingCreate operations. The special subject AllAuthenticated is the default policy for this role.

Note: CosNaming roles are only effective when administrative security is enabled.

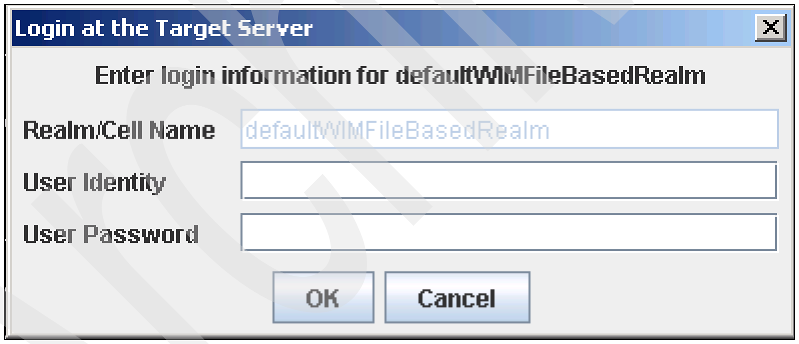
2.7.1 Mapping a user or a group to a CosNaming role

The process of mapping a user or group to a CosNaming role is similar to mapping a user or a group to an administrative role. To map CosNaming roles, click **Environment** → **Naming** → **CORBA Naming Service Users** for user mappings and **Environment** → **Naming** → **CORBA Naming Service Groups** for group mappings.

2.7.2 Applying CosNaming security: An example

This section shows a simple, practical example of the use of CosNaming security. WebSphere Application Server provides a Java application client `<`*`WebSphere_home`*`>\bin\dumpNameSpace.bat`, which is useful for listing all of the CORBA naming services that are available in the server.

When running `dumpNameSpace.bat` in a secure WebSphere environment, you are prompted with a window similar to the window that is shown in Figure 2-16.


Login at the Target Server

Enter login information for defaultWIMFileBasedRealm

Realm/Cell Name: defaultWIMFileBasedRealm

User Identity:

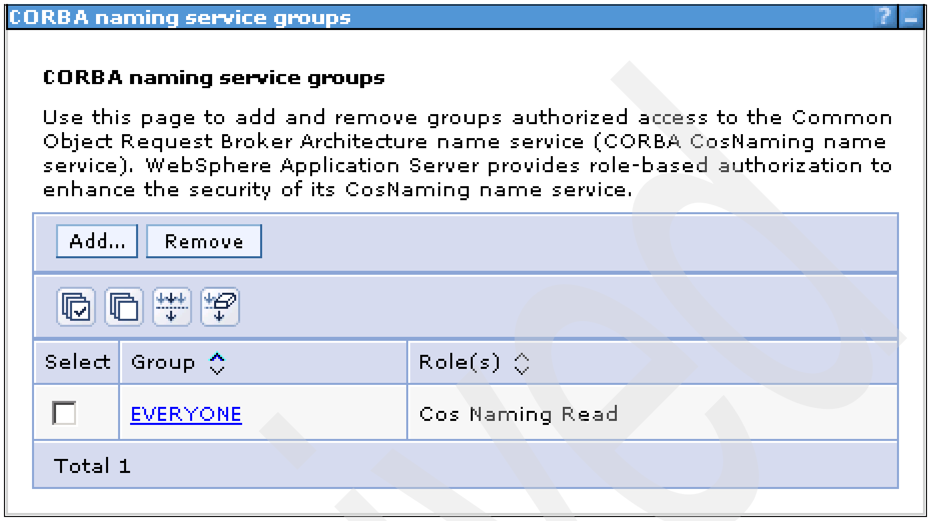
User Password:

Figure 2-16 A window prompted by the dumpNameSpace.bat Java application client

The window that is shown in Figure 2-16 is shown when the property `com.ibm.CORBA.loginSource` is the default value "`prompt`" in the CORBA client configuration file `sas.client.props`.

You can either fill in *any* correct user ID and password defined in your user registry and click **OK**, or you can just simply click **Cancel**.

With a default setup of the WebSphere Application Server, both actions will run without a problem, because the CosNaming read rights role is valid for everyone. Refer to Figure 2-17.



CORBA naming service groups

Use this page to add and remove groups authorized access to the Common Object Request Broker Architecture name service (CORBA CosNaming name service). WebSphere Application Server provides role-based authorization to enhance the security of its CosNaming name service.

Select	Group	Role(s)
☐	EVERYONE	Cos Naming Read

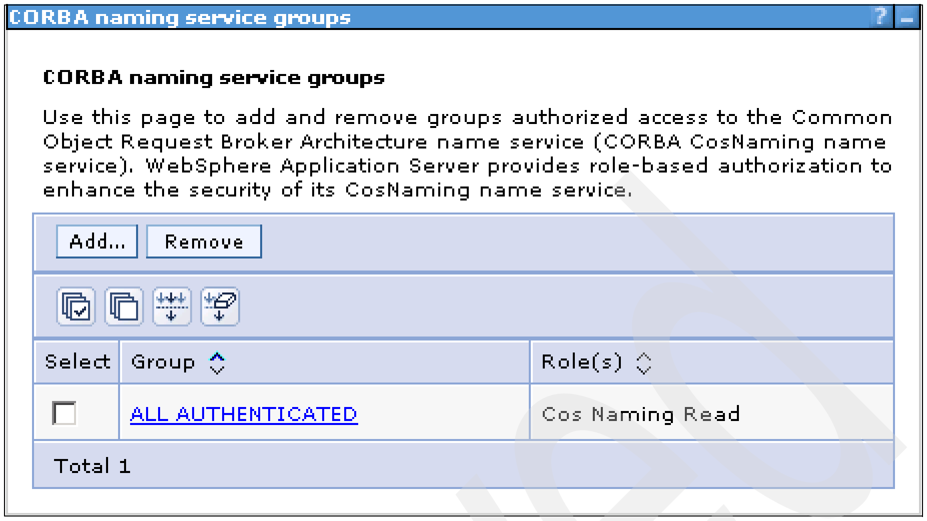
Total 1

Figure 2-17 Default CosNaming security for WebSphere Application Server

The following process shows a simple example of how to restrict the access to the CORBA naming service by allowing read access only to authenticated users:

1. From the administrative console, click **Environment** → **Naming** → **CORBA Naming Service Groups**.
2. Remove the entry for the special role group `EVERYONE`.
3. Add a new entry giving Cos Naming Read rights for the special group `ALL_AUTHENTICATED`.

   The final setup for the CosNaming security must be as shown in Figure 2-18 on page 49.


CORBA naming service groups

CORBA naming service groups

Use this page to add and remove groups authorized access to the Common Object Request Broker Architecture name service (CORBA CosNaming name service). WebSphere Application Server provides role-based authorization to enhance the security of its CosNaming name service.

Add... Remove

Select	Group	Role(s)
☐	ALL AUTHENTICATED	Cos Naming Read

Total 1

Figure 2-18 Customized CosNaming security

4. Save the setup and restart the WebSphere Application Server.

After the server has been started, running the `dumpNameSpace.bat` Java application client only works if you enter a correct user ID and password during the authentication process. Otherwise, the WebSphere Application Server throws an exception:

```
org.omg.CORBA.NO_PERMISSION
```

Important: Granting read access to `EVERYONE` presents a small security risk; therefore, it is better to keep the CosNaming security settings as shown in Figure 2-18. If you experience unexpected results in applications that use the CORBA naming service that you cannot resolve with application security roles, add the default CosNaming security entry back to the configuration as shown in Figure 2-17 on page 48. This security risk can be mitigated by ensuring that your WebSphere Application Server infrastructure is protected from other systems by firewalls.

3 Using security domains

Security settings for WebSphere Application server processes are defined in the *global security settings*. Global security is specific to one cell and is applicable cell-wide.

With WebSphere Application Server V7, you have the ability to define additional security domains that can override a subset of the global security settings. These domains can be used to provide customized security settings for applications and service integration buses. A security domain has a scope that defines where its settings are applicable. Settings that are not defined in the domain default to the global security settings.

This chapter describes security domains for applications. It contains the following topics:

- "Global security compared to security domains" on page 52
- "Application security domain scenarios" on page 54

3.1 Global security compared to security domains

The global security domain in WebSphere Application Server V7 defines the administrative security configuration and the default configuration for applications. If no other security domains are configured, and application security is enabled at the global security domain, all of the user applications and administrative applications use the same security configuration.

Although extremely convenient and straightforward, a single-domain configuration might not be the ideal configuration for certain clients that need settings customized for applications. Fortunately, WebSphere Application Server V7 offers the flexibility to override the global security domain configuration with additional security domains that are configured at a different scope. Security domains provide the flexibility to use configuration security settings that differ from those settings that are specified in the global security settings.

Administrative security must be enabled before you can enable application security. However, application security can be disabled at the global security level and enabled at the security domain level.

Note for V6.1 users migrating to V7:

In WebSphere Application Server V6.1, it was possible to use server-level security to customize security to a certain extent to override the global security settings. This feature was commonly used to enable or disable application security or Remote Method Invocation (RMI)/Internet Inter-ORB Protocol (IIOP) security on the application server level. There were limitations though; for example, it did not provide the capability to configure a different authentication or user registry on an individual server basis.

If you used server-level security in previous releases of WebSphere Application Server, you need to now use multiple security domains instead. Server-level security is deprecated in this release.

3.1.1 Attributes that can be configured in a security domain

You define attributes at the security domain level that need to be different from those at the global level. If the information is common, the security domain does not need to have the information duplicated in it. Any attributes that are missing in the domain are obtained from the global configuration.

Table 3-1 shows a comparison of the security features that can be specified in the global security settings and those that a security domain can override.

Table 3-1 Comparison of global and domain security settings

Global security configuration	Security domain overrides
► Enablement of application security ► Java 2 security ► User realm (registry) ► Trust Association Interceptor (TAI) ► SPNEGO Web authentication ► RMI/IIOP Security (CSIv2 protocol) ► JAAS ► Authentication mechanism attributes ► Authorization provider ► Custom properties ► Web attributes (single sign-on) ► SSL ► Audit ► LTPA authentication mechanism ► Kerberos authentication mechanism	► Enablement of application security ► Java 2 security ► User realm (registry) ► Trust Association Interceptor (TAI) ► SPNEGO Web authentication ► RMI/IIOP Security (CSIv2 protocol) ► JAAS ► Authentication mechanism attributes ► Authorization provider ► Custom properties

3.1.2 Configuration files

The global security configuration data is stored in the `security.xml` file, which is located in the `*profile_home*/cells/*cell_name*` directory.

For every security domain that is configured, two files are created in the `*profile_home*/config/waspolicies/default/securitydomains/*domain_name*` directory:

- The `security-domain.xml` file, which contains one or more security attributes, such as user registry, Java 2 security, authentication, JAAS login modules, TAI, and so forth
- The `security-domain-map.xml` file, which contains the scope of the security domain

Normally, security domain attributes override the user realm defined in the `security.xml` file. However, there are two exceptions:

- JAAS application logins, JAAS system logins, and JAAS J2C authentication data defined at the domain level are merged with the attributes defined at the global security domain.
- Custom properties defined in a security domain are also merged with those custom properties at the global security domain.

3.1.3 Security domain scope

A security domain can be scoped to an entire cell, or to a specific set of servers, clusters, or service integration buses. Therefore, multiple security domains can be used to allow security settings to vary from one application to another application.

Security settings that apply to an application will be defined by the following scope:

1. If the application is running on a server or cluster that is within the scope of a security domain, those settings will be used. Security settings that are not defined in this domain will be taken from the global security settings (not a cell-level domain).
2. If the application is running on a server or cluster that is not within the scope of a security domain, but a security domain has been defined at the cell scope, that domain will be used. Security settings that are not defined in this domain will be taken from the global security settings.
3. If the previous conditions do not apply, the global domain settings will be used.

Note that you can enable or disable application security at the domain and global level, so just falling within a domain does not necessarily mean that application security is enabled. Also, note that naming operations always use the global security configuration.

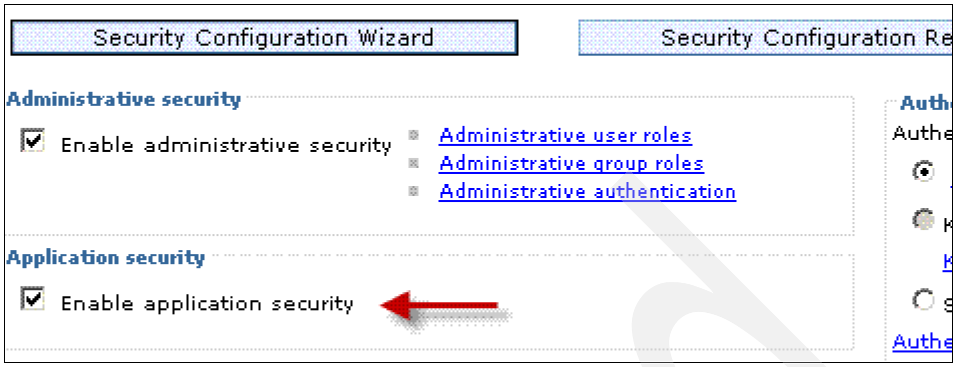
3.2 Application security domain scenarios

Security domains can be created and managed using the administrative console or scripts. Only users assigned to the administrator roles can configure security domains.

The following sections discuss two simple scenarios to illustrate the concept.

3.2.1 Scenario: Application security at the global security level

If all applications can be secured using a shared user realm, you can simply define security at the global domain level. The global settings are found by navigating to **Security** → **Global security** in the administrative console (Figure 3-1 on page 55). Select **Enable application security** and define the user registry and other security settings as required.


Security Configuration Wizard

Security Configuration Re

Administrative security

☑ Enable administrative security

- Administrative user roles
- Administrative group roles
- Administrative authentication

Application security

☑ Enable application security

Auth

Authe

Authe

Figure 3-1   Enable application security at the global security domain

If no other security domains are configured to override the global security domain, all of the applications use information from the global security domain configuration. Applications will share the same user realm, because only one active user registry is allowed at the global security level. The predefined user registry is the file-based federated repository utilizing a simple `fileRegistry.xml` file.

3.2.2 Scenario: Security domains that override global security

In this example, administrative security is enabled and uses a federated repository for authorization. Application security is not enabled at the global domain level. A security domain is created to enable application security for all applications running on the application server, server1. The local OS user registry is used as the user realm for the new security domain.

To create this environment, a new security domain called TestDomain is created with the local OS defined as the user registry.

The administrative applications, as well as the naming operations in that scope, will still use the global security configuration.

Note: The local operating system user registry is used here for demonstration purposes. The steps are extremely similar if other user registries, such as Lightweight Directory Access Protocol (LDAP), are chosen.

Step 1: Configure local operating system user registry

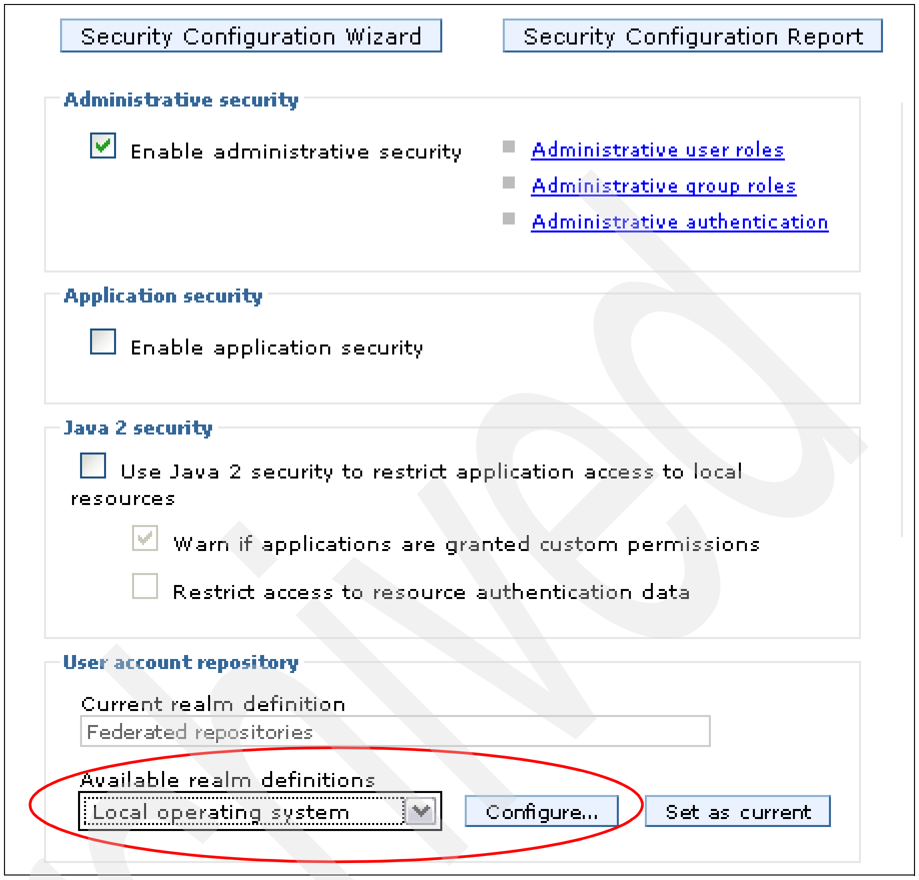
The first step in creating this environment is to configure the local operating system user registry:

1. Start by ensuring that the deployment manager and application server server1 are running.
2. Open the administrative console, and verify that administrative security is enabled.
3. Create a local operating system user ID. For example, on Windows, open a command window and execute the following command:

   ```
   net user wslocalos wslocalos /add
   ```

4. Add this new user ID to the local operating system user registry. The new user ID will be used to access the applications that run on servers in the domain. In this example, the user ID is `wslocalos`.
5. Configure the new user registry to WebSphere Application Server:
   a. In the administrative console, navigate to **Security** → **Global security**.
   b. Select **Local operating system** from the Available realm definitions pull-down (Figure 3-2 on page 57) and click **Configure**.

      Note that the current realm for the global security domain is set to Federated repositories. Configuring another realm does not change this setting unless you use “Set as current.” In this case, the local operating system realm is being configured for use later when a new security domain is defined.


Security Configuration Wizard | Security Configuration Report

Administrative security

[✓] Enable administrative security

- Administrative user roles
- Administrative group roles
- Administrative authentication

Application security

[ ] Enable application security

Java 2 security

[ ] Use Java 2 security to restrict application access to local resources

[✓] Warn if applications are granted custom permissions

[ ] Restrict access to resource authentication data

User account repository

Current realm definition: Federated repositories

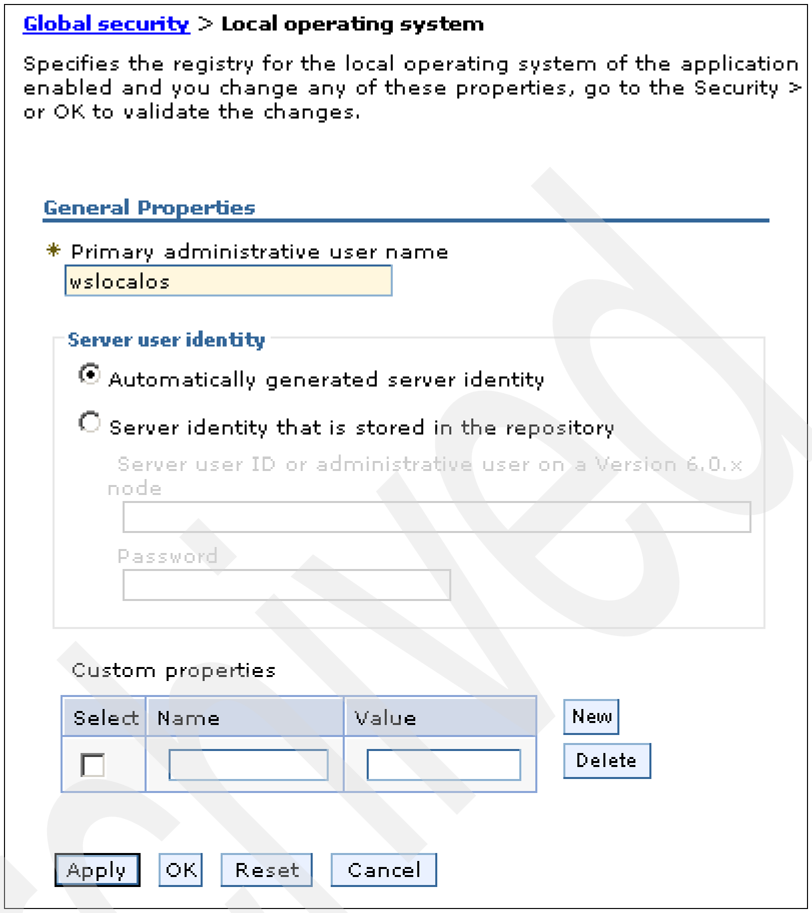
Available realm definitions: Local operating system

Configure... | Set as current

Figure 3-2 Configure Local operating system user registry

c. In the next panel (Figure 3-3 on page 58), enter the primary administrative user name for this security domain.

Specify the name of a user that is defined in your local operating system. The user name is used to log on to the administrative console when administrative security is enabled. Note that configuring the registry is a separate act from configuring a domain to use it. So, even though this registry will not be used for administrative security, you still must assign a name here.


Global security > Local operating system

Specifies the registry for the local operating system of the application
enabled and you change any of these properties, go to the Security >
or OK to validate the changes.

General Properties

* Primary administrative user name
wslocalos

Server user identity

(●) Automatically generated server identity

( ) Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

Custom properties

Select	Name	Value
☐		

Figure 3-3 Local OS user registry settings

The server identity is used for communications between servers. Normally, you allow this server identity to be automatically generated. However, if you are adding a V5 or V6 server to the domain, you must make sure that the server identity is defined in the user repository and specified here. This option does not apply to z/OS or i5/OS® platforms.

Click **OK**.

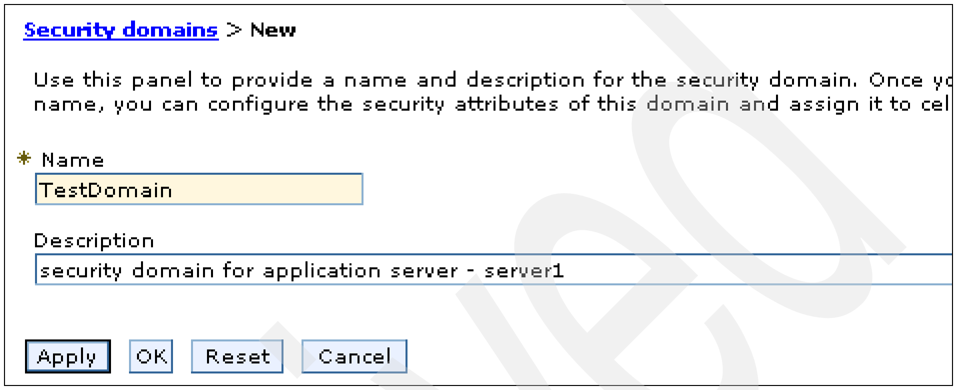
d. Save the changes.

Step 2: Create a new security domain

The next step is to create the security domain that enables application security for server1 and specifies the local operating system as the user registry.

To create the new security domain, perform the following steps:

1. In the administrative console, select **Security** → **Security domains**. Click **New** to create the new security domain.
2. Enter `TestDomain` for the name, and enter an appropriate description (Figure 3-4).



Security domains > New

Use this panel to provide a name and description for the security domain. Once yo
name, you can configure the security attributes of this domain and assign it to cel

Name: TestDomain

Description: security domain for application server - server1

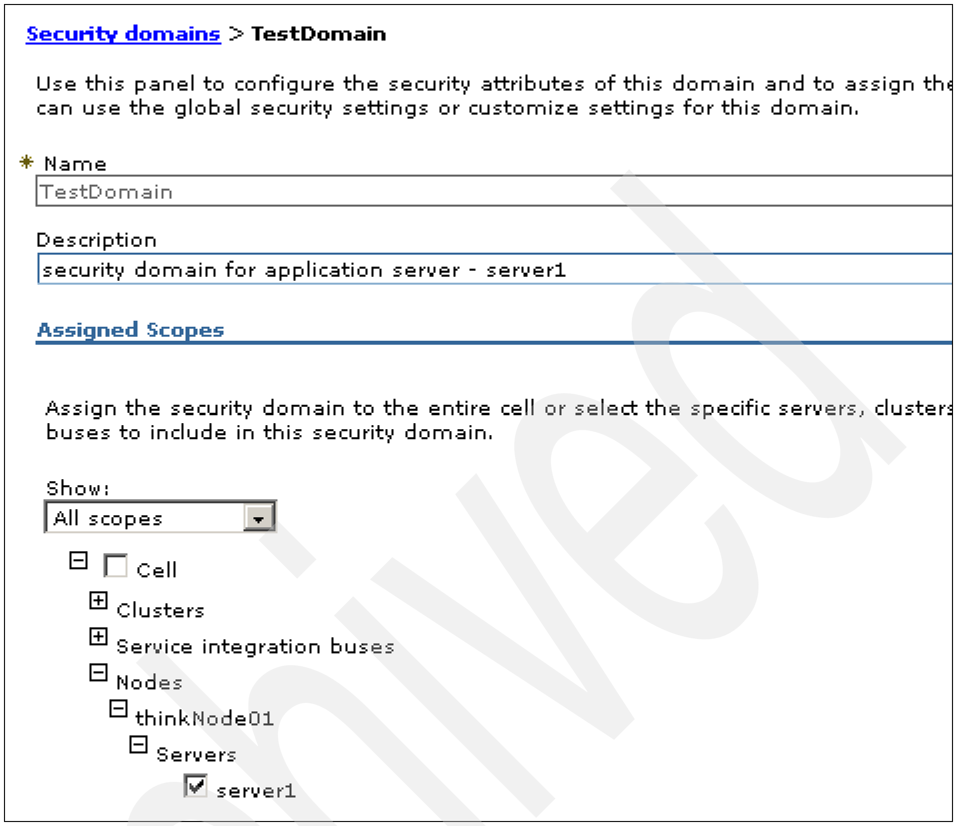
Figure 3-4 Create a new security domain

3. Click **OK** and save the changes.
4. Click **TestDomain** in the list of security domains to open the configuration page. The configuration page for the security domain has three major sections. The domain name and description are in the first section.

   The next section, Assigned Scopes, contains the settings that allow you to assign the scope for the domain. All servers, clusters, or service integration buses that are selected here will use the settings in this security domain. Assigning a domain to the cell scope effectively creates a default configuration for application security.

   In this example, we select **Server1** as the scope for the domain.

   These two sections are shown in Figure 3-5 on page 60.


Security domains > TestDomain

Use this panel to configure the security attributes of this domain and to assign the
can use the global security settings or customize settings for this domain.

* Name
TestDomain

Description
security domain for application server - server1

Assigned Scopes

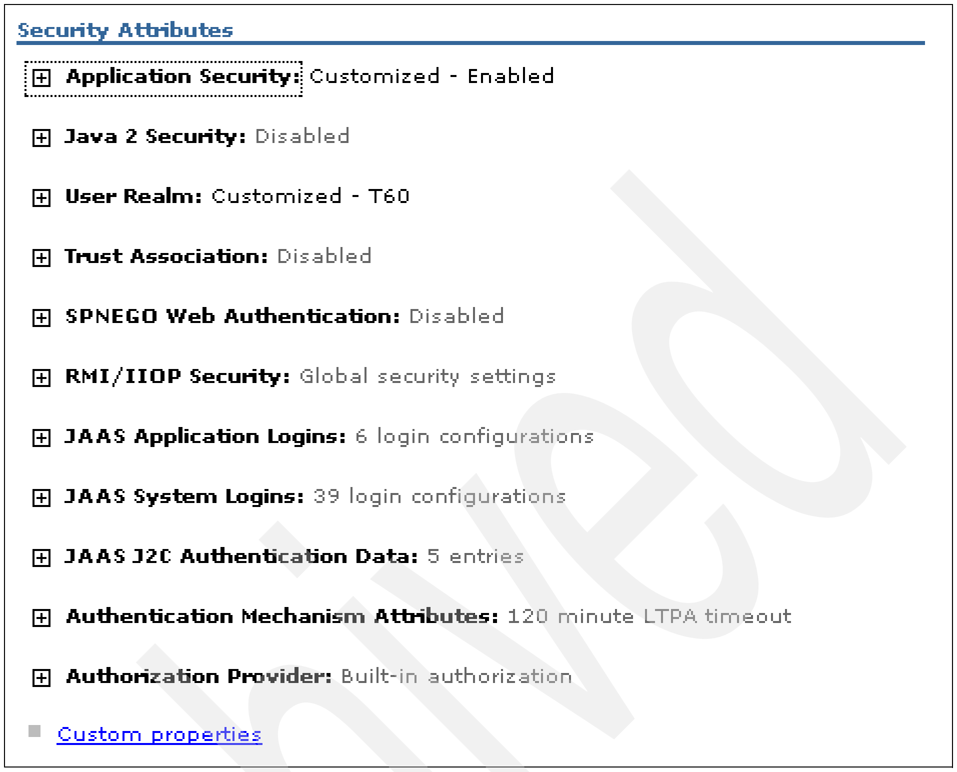
Assign the security domain to the entire cell or select the specific servers, clusters
buses to include in this security domain.

Show:
All scopes

- ☐ Cell
  - Clusters
  - Service integration buses
  - Nodes
    - thinkNode01
      - Servers
        - ☑ server1

Figure 3-5 Configure a new Security Domain

5. The last section contains expandable sections for each type of security attribute that you can configure in the domain (Figure 3-6 on page 61).


Security Attributes

⊞ **Application Security:** Customized - Enabled

⊞ **Java 2 Security:** Disabled

⊞ **User Realm:** Customized - T60

⊞ **Trust Association:** Disabled

⊞ **SPNEGO Web Authentication:** Disabled

⊞ **RMI/IIOP Security:** Global security settings

⊞ **JAAS Application Logins:** 6 login configurations

⊞ **JAAS System Logins:** 39 login configurations

⊞ **JAAS J2C Authentication Data:** 5 entries

⊞ **Authentication Mechanism Attributes:** 120 minute LTPA timeout

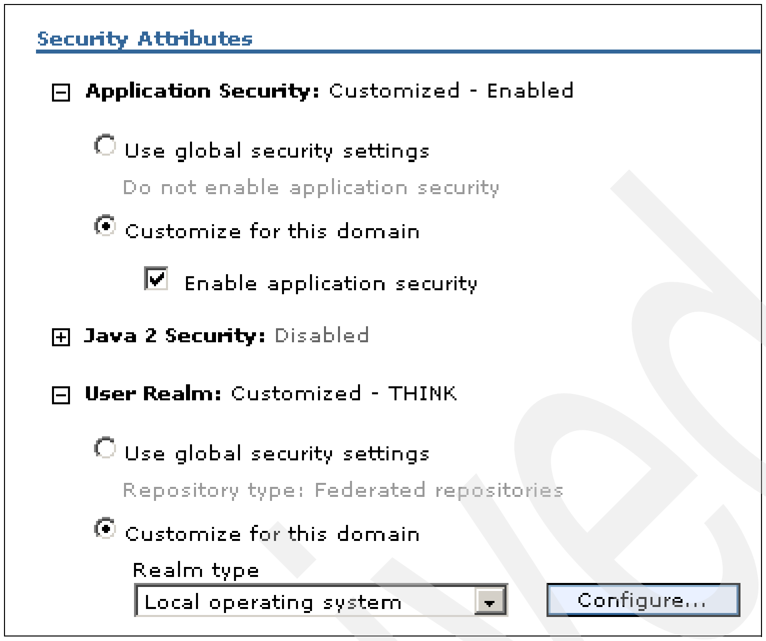
⊞ **Authorization Provider:** Built-in authorization

■ Custom properties

Figure 3-6 Security attributes section of the security domain configuration page

6. Under the Security Attributes section, expand **Application Security**.

   In this example, we enable application security for this domain by selecting **Customize for this domain** and **Enable application security** (Figure 3-7).


Security Attributes

⊟ **Application Security:** Customized - Enabled

○ Use global security settings

Do not enable application security

● Customize for this domain

☑ Enable application security

⊞ **Java 2 Security:** Disabled

⊟ **User Realm:** Customized - THINK

○ Use global security settings

Repository type: Federated repositories

● Customize for this domain

Realm type: Local operating system

Figure 3-7 User realm settings of a security domain

7. Next, expand the **User Realm** portion. Ensure that **Customize for this domain** is selected, and then, select **Local operating system** from the pull-down list (Figure 3-7).
8. Click **OK** and save the changes.
9. Restart the application server for the changes to take effect.

Step 3: Test the new security domain

In our test environment, we ran a simple test to illustrate that the user registry that is used for administrative security was still the file-based repository and that the applications on server1 used the local OS. You do not have to test for this situation; however, make sure that your applications are secured using the registry that is specified for the domain. Verify that you set the scope for the domain correctly and that the user registry is populated with the correct user IDs.

This scenario created a new domain that uses a separate user registry (local OS) than the registry defined in the global security (federated repositories).

You can simply test to ensure that the the user registries have been properly defined by attempting to access both the administrative functions and the application with the user IDs with which you expect to have access.

This simple test assumes that:

- The administrator user ID, wasadmin, is only defined in the federated repositories.
- The user ID that is required to access the application, wslocalos, is only defined in the local operating system.

Test the administrator user IDs

With the new security domain configured, the user ID for the administrator (`wasadmin`) is the same as before, because the administrator user IDs defined in the file-based federated user repository are still being used by the global security settings.

The goal of this test is to ensure that the administrator user IDs can still have authority to perform administrative functions (meaning, the file-based repository is still used when administrative functions are accessed). However, trying to authenticate to the console using wslocalos will fail , because that user is not defined in the file-based federated user repository (and it must also be further defined as a console user).

To ensure that the administrative security is still working properly:

1. Restart the deployment manager and application server server1.
2. Log in to the administrative console using the wasadmin user. The login will succeed.
3. Log out from the console and try logging in as wslocalos, which fails because wslocalos does not exist in the federated repository.

When administrators create a security domain and associate it with a scope, only the user applications in that scope use the security attributes that are defined in the security domain.

Test access to the application

Because the new security domain enabled the application security for server1, accessing the snoop servlet on that server requires the user to authenticate.

Using the wasadmin user ID for authentication to snoop will fail, because it is not defined in the registry that is configured as part of the security domain (local operating system registry).

Authenticating to snoop using the wslocalos user ID succeeds, because that user ID exists in the local operating system registry.

4

Configuring the user registry and authentication settings

This chapter provides an introduction to configuring user registries. WebSphere Application Server uses registries to authenticate users and to retrieve users and groups to create a map of security roles that are used to authorize user actions.

This chapter describes the approach for configuring registries for use by a WebSphere Application Server cell. It contains the following topics:

- “User registry basics” on page 66
- “Configuring a stand-alone LDAP registry” on page 69
- “Federated repositories” on page 95
- “Authentication and authorization settings” on page 133

4.1 User registry basics

A *user registry* is an abstraction that is used by WebSphere Application Server to standardize the term that is used to represent the various implementations of user and group repositories that the application server can be configured to use.

4.1.1 User registry types

WebSphere Application Server can be configured to use the following registry implementations:

- Local operating system (Local OS)

  For Local OS, the users and user groups are retrieved from the operating system. This implementation is the typical practice when using z/OS.

- Federated repositories

  Federated repositories support the configuration of one or more user repositories for the purpose of providing a unified view of the user and group information that is owned by each repository. Federated repositories support file-based, LDAP, database, and custom registry implementations. When administrative security is enabled during profile creation, a federated repository with a file-based registry is created to hold the administrator user IDs.

  We discuss this configuration approach further in 4.3, "Federated repositories" on page 95.

- Stand-alone LDAP

  WebSphere Application Server can connect to a stand-alone LDAP directory server using the LDAP protocol. If you are configuring for a single registry, we recommend this type of registry for use with distributed platform environments.

  We discuss configuring a stand-alone LDAP in 4.2, "Configuring a stand-alone LDAP registry" on page 69.

- Stand-alone custom registry implementation

  Custom registries can be written and integrated with WebSphere Application Server. You can obtain information about writing a custom registry at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_customauth.html`

  WebSphere provides a custom user registry example that is not meant for production, but this example shows the application programming interfaces (APIs) that need to be implemented if a custom registry is used.

Implementing custom security that is actually secure is not easy, and for this reason, we do not recommend custom registry implementations. However, if you are considering this option, be aware that:

- Custom registries must not rely on application server services, such as Enterprise JavaBeans.
- The custom interface for stand-alone registries is a different interface that is used by federated repositories. A custom stand-alone registry cannot be federated without first being modified to implement the federated repository service provider programming interface (SPI).
- The interface must be completely implemented, including error scenarios.
- Availability and failover will need to be considered.
- The custom registry implementation alone is not the only custom code that is needed if products are used that build on the foundation of WebSphere Application Server, for example, WebSphere Portal or WebSphere Process Server. These products have their own security extensions that must be considered in addition to those security extensions that are satisfied by the stand-alone custom registry interface.
- We recommend that any custom implementations undergo rigorous third-party penetration testing to test for security vulnerabilities.

Best practice: With the possible exception of z/OS, LDAP directories are the repository of choice. While repository types other than LDAP are supported, only LDAP is the recognized industry standard.

4.1.2 User registry content

A user registry provides the application server with user and group information for mapping with Java Platform, Enterprise Edition (JEE) security roles. Groups are the method that is used by a registry to represent users that share a common function or attribute.

Best practice: When designing the security access model for your application server cell, we recommend that all access relationships are mapped to groups in the registry in preference to individual users, even if the group only has one user. This approach decouples individual users from applications and administrative functions in the cell, making security administration significantly more manageable.

4.1.3 Using multiple registries with domains

When configuring a WebSphere Application Server cell, it is possible to have multiple user registries configured and active within the cell, because registry implementations can be independently configured and linked to security domains.

In a cell, there is a default global security scope. But there can also be other security domains defined within the cell. These separate security domains can be configured to have separate registries.

Note: While each security domain can have its own registry, it is only possible for a federated repository to exist at the global cell domain scope. Registries configured for other security domains are stand-alone registry configurations.

Each registry configuration is provided a name that is either defaulted by the application server environment or explicitly named when the registry is configured. This name is called the *user realm* for the registry.

Figure 4-1 on page 69 shows an example that illustrates two security realms defined in a cell. In this example, a specific security domain has been defined for the organization’s partner applications.

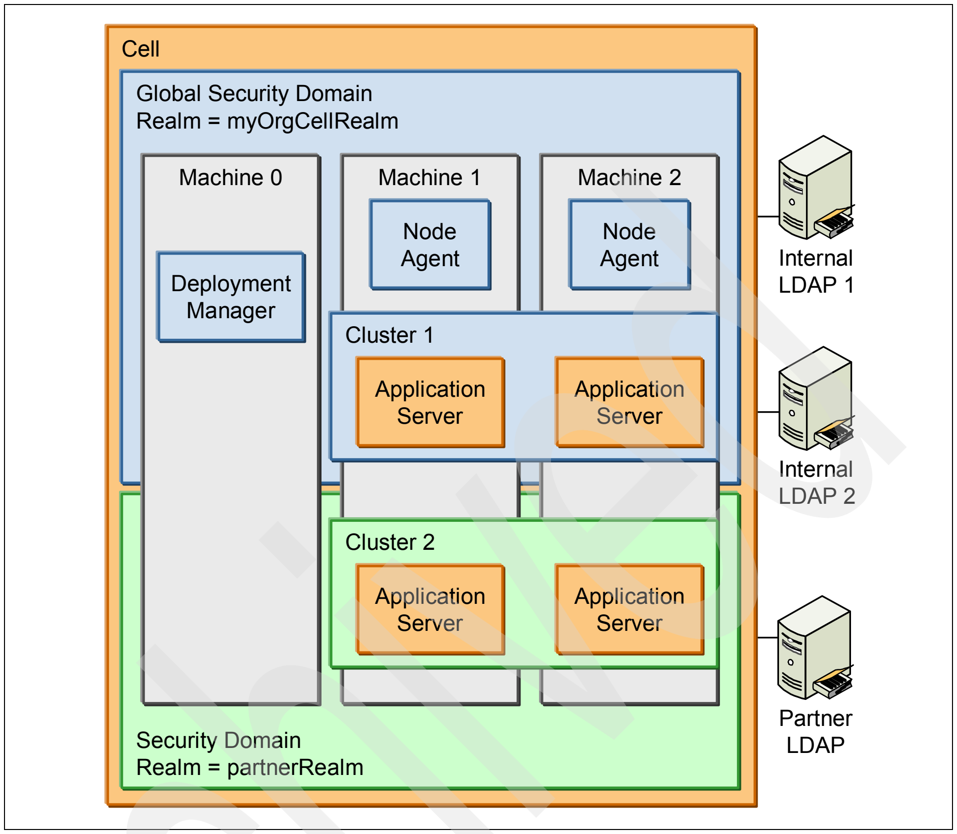


Figure 4-1 Simple multiple user realm illustration

We provide specific discussions about security domains in Chapter 3, “Using security domains” on page 51. For the purposes of understanding the user registry relationship, configuring a security registry for a domain is the same process as configuring a stand-alone registry at the global security level.

4.2 Configuring a stand-alone LDAP registry

Before configuring WebSphere Application Server to use a stand-alone LDAP as the user registry, the LDAP directory must be separately installed and configured.

In most cases, organizations have established registries of user information. If your organization is only starting with LDAP directories, we recommend that an experienced security consulting group is engaged to assist with the directory design. References, such as *Understanding LDAP - Design and Implementation*, SG24-4986, and the LDAP directory documentation can also be of assistance.

For information about supported LDAP directories with WebSphere Application Version 7, refer to the IBM Support "List of supported software for WebSphere Application Server V7.0" at:

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369

While LDAP as a protocol is a widely accepted standard, each vendor's directory implementation typically uses different directory structures and object classes for users and groups for communicating with a directory. Thus, while the steps for configuring a stand-alone LDAP are the same, the LDAP query mapping for object classes to users and groups in the directory tends to vary from implementation to implementation.

In the following sections, we illustrate examples of configuring a stand-alone LDAP by using an example directory configured in IBM Tivoli Directory Server.

4.2.1 Configuration checklist

The checklist in Table 4-1 on page 71 is an abbreviated set of steps that can be used in planning the configuration of the stand-alone LDAP registry.

Table 4-1 Configuration checklist

Check item	Description	Reference
1	Verify that the chosen LDAP directory is in the compatible software list.	http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369
2	Configure LDAP to communicate using Secure Sockets Layer (SSL).	5.4, "Advanced concepts" on page 184
3	Identify general property configurations: - primary admin ID - Host - Port - Base distinguished name (DN) - Bind DN - Bind password	4.2.2, "Understanding the directory structure" on page 71 Figure 4-7 on page 78
4	Identify object class filters for user and group filters.	4.2.2, "Understanding the directory structure" on page 71 Figure 4-5 on page 76 Figure 4-6 on page 77
5	Choose authentication and authorization mechanisms that might include: - Configure SSO. - Require SSL for RMI/IIOP. - Identify any realms that might need to be trusted.	4.4, "Authentication and authorization settings" on page 133

4.2.2 Understanding the directory structure

It is important that you understand the design of the LDAP registry that you will use. One of the easiest ways to understand the design is to utilize a directory information tree diagram. The directory information tree provides information about the structure being used for users and groups. But to make this directory information tree diagram even more useful for configuration purposes, include the directory object classes to make the configuration steps easier to understand in later steps. Figure 4-2 shows the directory information tree for the IBM Tivoli Directory Server that is being used in this configuration example.

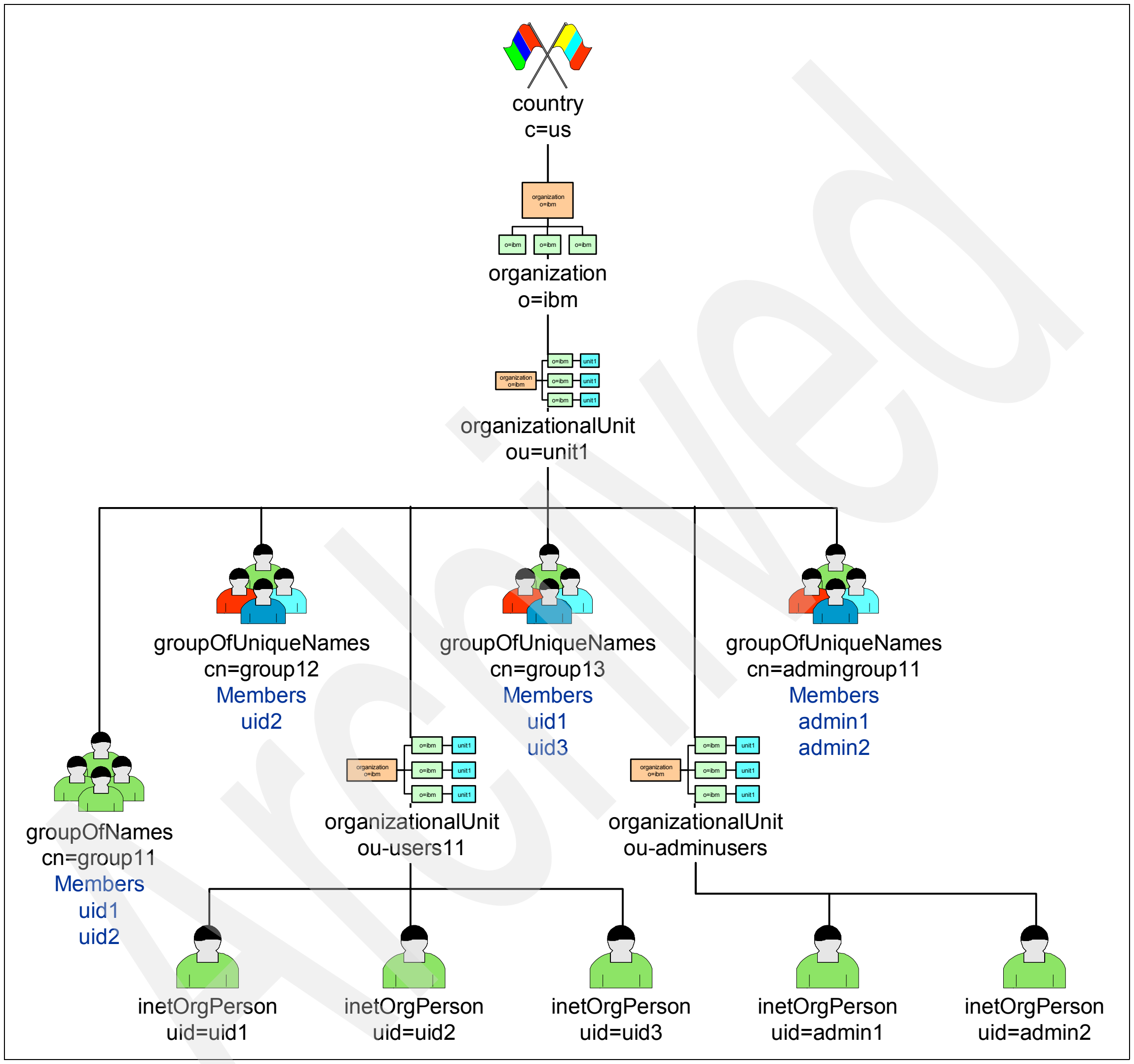


Figure 4-2 Directory information tree for the example LDAP directory

Important: This diagram illustrates the simple structure that is used for this example. This structure is intended to assist in the demonstration of how to configure elements of security in WebSphere Application Server. It is not intended to be a guide for the design of an LDAP directory. Such a design requires considerations specific to each organization.

Information that is important for configuring the LDAP registry is easy to follow in the diagram. From the diagram, the following relevant directory information is determined:

1. The base distinguished name for this directory is intended to be `ou=unit1,o=ibm,c=us`.
2. There are two types of group objects being used:
   - `groupOfNames`
   - `groupOfUniqueNames`
3. The groups use the "uid" attribute of the user objects to identify the unique distinguished names of users.
4. The type of object used for user objects is inetOrgPerson.
5. We recommend that you utilize group mappings. In this registry, there are four groups:
   - `group11`
   - `group12`
   - `group13`
   - `admingroup11`

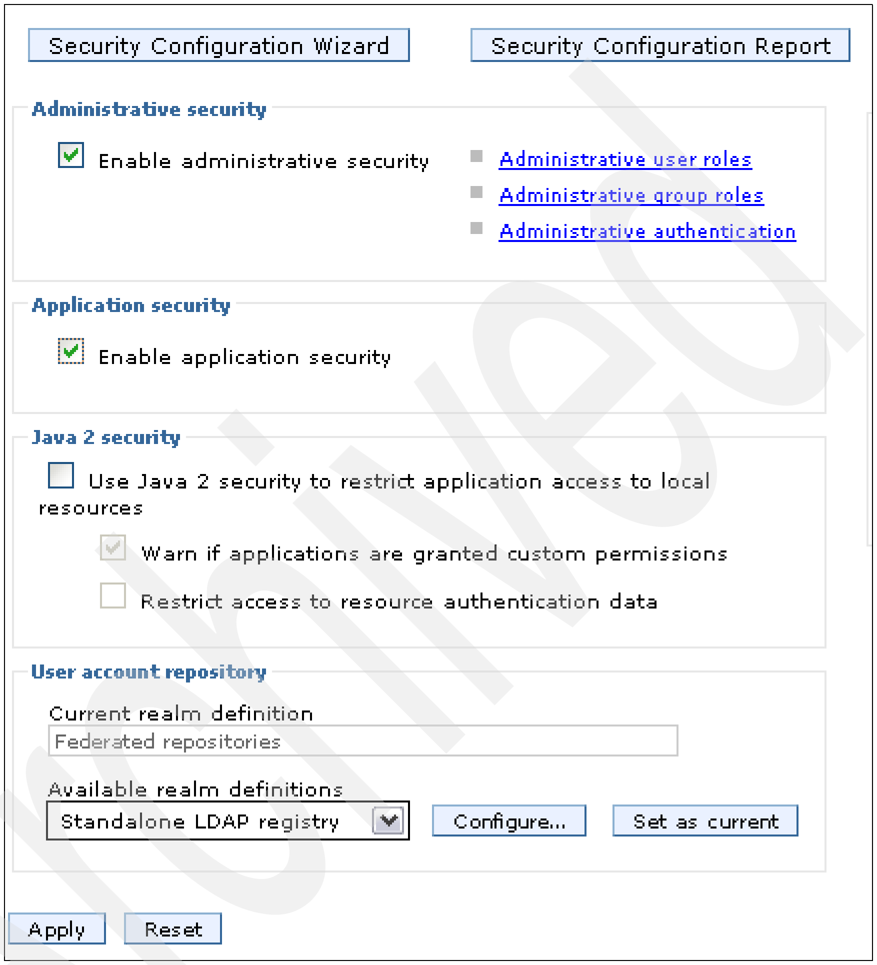
At the beginning of this example, only static groups are used. The directory structure will be expanded later when we discuss dynamic and nested groups. In static groups, members are individually added and removed from the group, which means that it is a fixed list and the only way for the list to grow is for a new user to be added to the list explicitly.

4.2.3 Configuring a stand-alone LDAP using the console

As was discussed in 4.1, "User registry basics" on page 66, a user registry can be configured for the global security domain, which is the security domain that we will configure in the example in the following pages:

1. Open the administrative console, and select **Security** → **Global security** to navigate to the Global security panel (Figure 4-3 on page 74).

2. In the User account repository section of the panel, select **Standalone LDAP registry** from the Available realm definitions drop-down list (Figure 4-3 on page 74).


Security Configuration Wizard
Security Configuration Report
Administrative user roles
Administrative group roles
Administrative authentication
Configure...
Set as current
Apply
Reset

Administrative security

✓ Enable administrative security

Application security

✓ Enable application security

Java 2 security

☐ Use Java 2 security to restrict application access to local resources

✓ Warn if applications are granted custom permissions

☐ Restrict access to resource authentication data

User account repository

Current realm definition: Federated repositories

Available realm definitions: Standalone LDAP registry

Figure 4-3 Global security

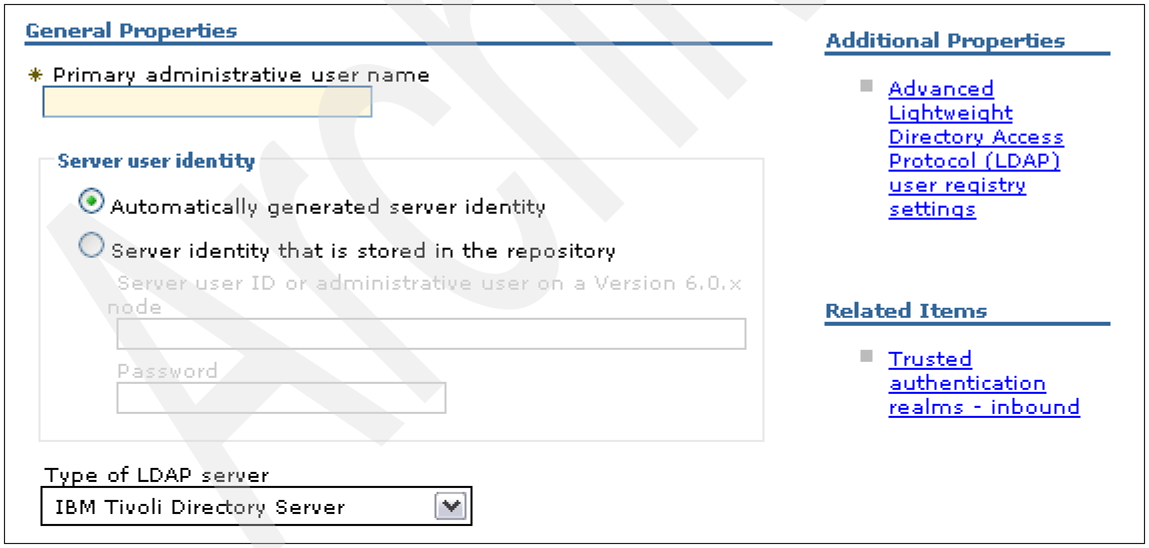
3. Click **Configure**.

Tip: Alternatively, you can click Security Configuration Wizard (shown in Figure 4-3 on page 74).

In the following steps, we will show that the directory information tree in this example does not use the WebSphere Application Server default object class types for IBM Tivoli Directory Server. The configuration wizard does not offer the option to configure the object classes. The wizard assumes a set of defaults based on the directory type. The wizard expects that the directory is configured with users and groups using certain object classes; therefore, if this situation is not true for your user object class type, the wizard cannot complete successfully.

Thus, if you know you are using non-default object classes or you are uncertain what is defaulted for your directory, it is better to use the extended configuration steps that are illustrated in this example. For more information about the various directory default object classes, refer to 4.2.6, "Stand-alone LDAP configuration defaults" on page 92.

4. Select the directory type from the Type of LDAP server drop-down list. If your directory server is not there, you will need to use the Custom option. (Figure 4-4).


General Properties

* Primary administrative user name

Server user identity

(●) Automatically generated server identity

( ) Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

Type of LDAP server: IBM Tivoli Directory Server

Additional Properties

- Advanced Lightweight Directory Access Protocol (LDAP) user registry settings

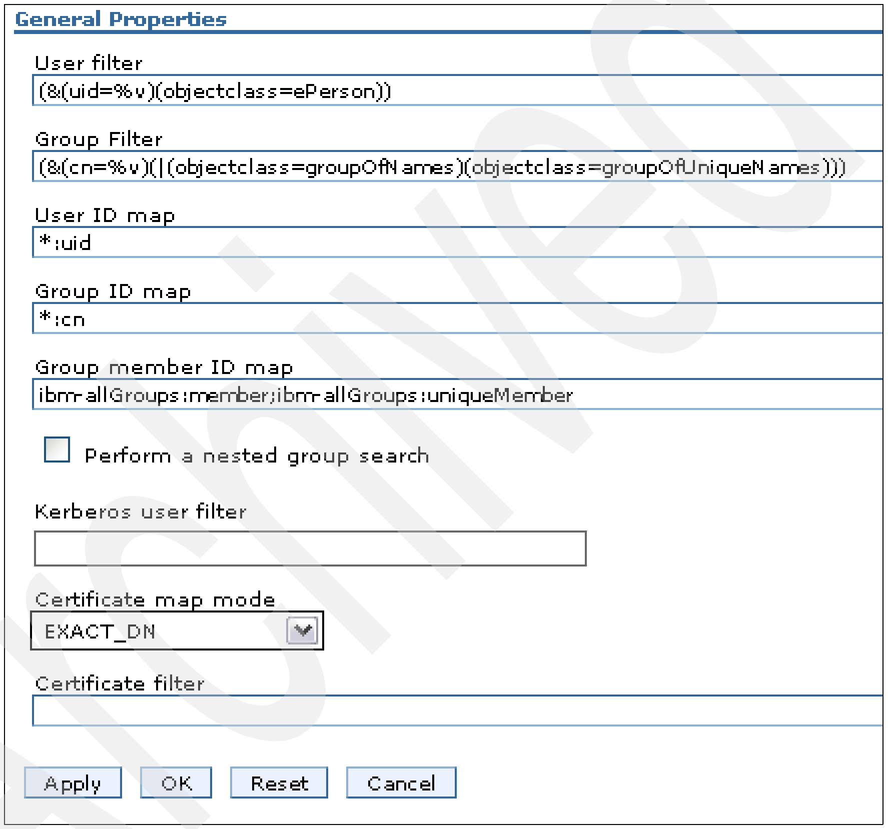
Related Items

- Trusted authentication realms - inbound

Figure 4-4 Global security → Standalone LDAP registry and select the LDAP type

The directory server chosen from the drop-down list changes the default object classes that are populated when executing the next step. In this example, **IBM Tivoli Directory Server** is selected. By selecting this directory type, the default object classes for this directory are populated into the LDAP query strings.

5. From the Additional Properties section of the window, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**. This navigation takes you to the panel that is shown in Figure 4-5.



General Properties

User filter: (&(uid=%v)(objectclass=ePerson))

Group Filter: (&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))

User ID map: *:uid

Group ID map: *:cn

Group member ID map: ibm-allGroups:member;ibm-allGroups:uniqueMember

☐ Perform a nested group search

Kerberos user filter:

Certificate map mode: EXACT_DN

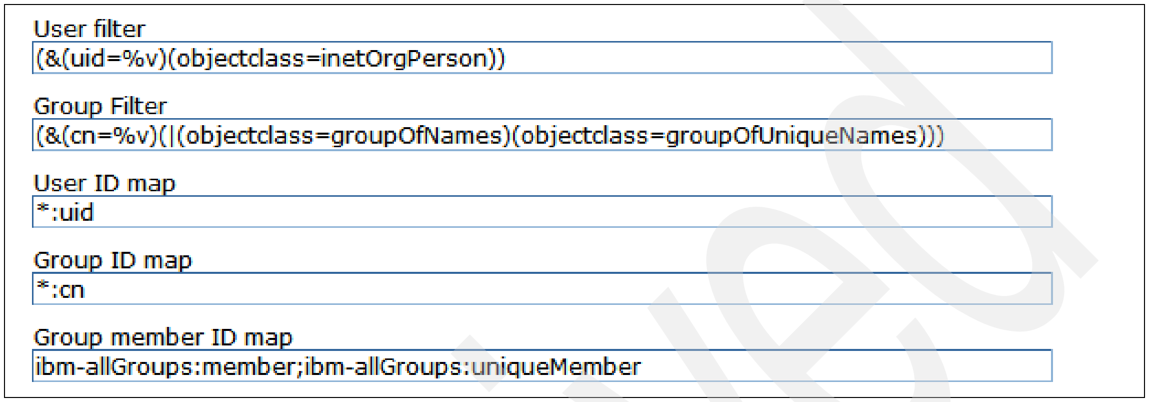
Certificate filter:

Figure 4-5 LDAP user registry settings: IBM Tivoli Directory Server defaults

6. Typically, you will be required to modify the object classes specified for the user and group filters. Subsequently, changes to the user ID and group ID mappings might also be needed depending on your directory configuration.

   These filters and ID mappings change how the application server queries the LDAP directory. In 4.2.2, “Understanding the directory structure” on page 71, it was determined that the directory is using an object class of inetOrgPerson for the users. So, the default object class of “ePerson” needs to be modified. It was also discussed that the registry is configured with static group types of groupOfNames and groupOfUniqueNames. These types are the default groups specified, so no modifications to the group filter are required.

Change the user filter as shown in Figure 4-6.



User filter
(&(uid=%v)(objectclass=inetOrgPerson))

Group Filter
(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))

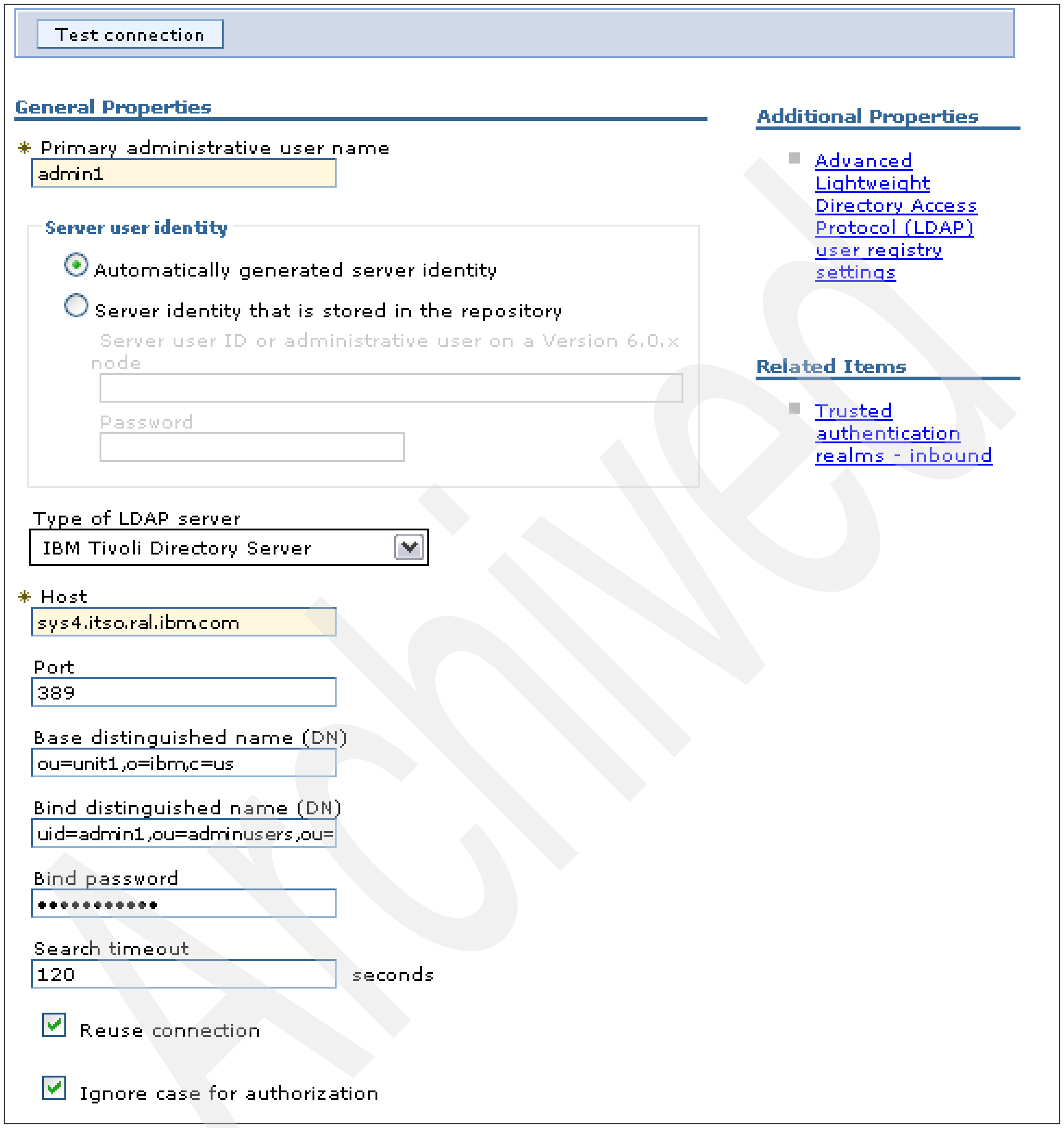
User ID map
*:uid

Group ID map
*:cn

Group member ID map
ibm-allGroups:member;ibm-allGroups:uniqueMember

Figure 4-6 Global security → Standalone LDAP registry → Advanced Lightweight Directory Access Protocol (LDAP) user registry: IBM Tivoli Directory modified user filter

7. Click **OK**, and save the configuration.
8. Enter the general properties for the LDAP configuration (Figure 4-7 on page 78).


Test connection

General Properties

* Primary administrative user name: admin1

Server user identity

[✓ Automatically generated server identity]

Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node:

Password:

Type of LDAP server: IBM Tivoli Directory Server

* Host: sys4.itso.ral.ibm.com

Port: 389

Base distinguished name (DN): ou=unit1,o=ibm,c=us

Bind distinguished name (DN): uid=admin1,ou=adminusers,ou=

Bind password: ●●●●●●●●●●●

Search timeout: 120 seconds

[✓] Reuse connection

[✓] Ignore case for authorization

Additional Properties

- Advanced Lightweight Directory Access Protocol (LDAP) user registry settings

Related Items

- Trusted authentication realms - inbound

Figure 4-7 Stand-alone LDAP registry: Configure general properties

The minimum general settings that need to be configured are:

– Primary administrative user name: `admin1`

– Host: `sys4.itso.ral.ibm.com`

- Base Distinguished name (identified from directory information tree): `ou=unit1,o=ibm,c=us`
- Bind distinguished name (the user that the application server uses to connect to the LDAP directory. In this case, `admin1` was arbitrarily chosen for this example): `uid=admin1,ou=adminusers,ou=unit1,o=ibm,c=us`
- Bind password: `admin1pwd01`
- Ignore case for authorization: For certain directories, this field is optional. When default authorization is active, this option allows authorization checks that are not case-sensitive. *You are required to select this option when using IBM Tivoli Directory Server*.

Optional general settings include:

- Reuse connection: We recommend that the reuse of connections is enabled; otherwise, the performance of the LDAP connectivity with the application server is likely to degrade significantly.
- SSL Settings: These settings are typically set for LDAP configurations. However for this book, you can read about configuring SSL information in Chapter 5, “Secure Sockets Layer administration” on page 151.

9. (Optional) Use **Test Connection** to test the connection.

Tip: The connection test connects to the host on the specified port and binds to the directory using the base DN, bind DN, and bind password. This test connection does not yet verify the successful search of the directory for the primary administration user ID using the user and group filters specified.

10. Click **OK**.
11. Click **Set as current** to make the LDAP directory the current user repository.
12. At this time, it is normal to configure any additional authentication and authorization preferences. We discuss these options further in “Authentication and authorization settings” on page 133. For the continuation of this example, assume these steps have been completed.
13. After finalizing your security preferences, click **Apply** on the Global security panel.

Important: Be sure to read the instructions that are supplied in the messages information regarding restarting and synchronizing nodes in a Network Deployment environment (shown in Figure 4-8 on page 81). These instructions remind you that in order to complete the configuration, you must synchronize and restart your cell.

After restarting the environment, you can authenticate using the primary administration ID and then proceed to configure administrative security (refer to Chapter 2, “Administrative security” on page 17).

Tip: If you are unable to authenticate, you can turn off security to help troubleshoot the problem by running the following **wsadmin** commands (command in jacl format).

Note that this command must be run locally; it cannot be executed remotely.

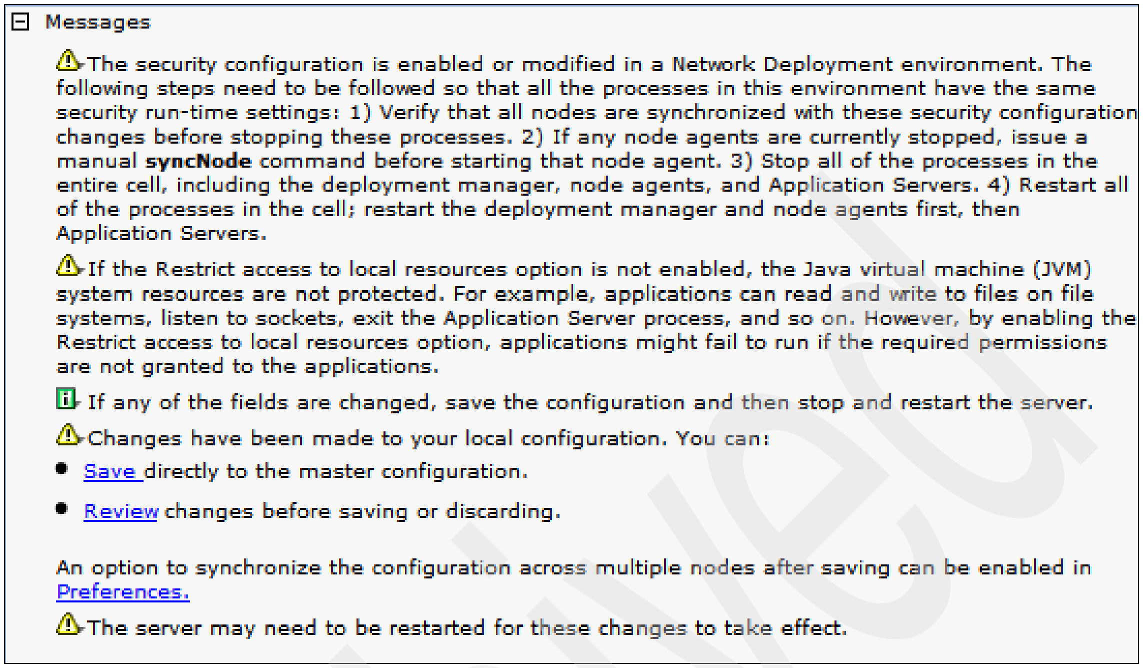
This command is executed from `<profile>/bin` *directory where* `<profile>` *is the profile's install directory.*

```
wsadmin -conntype NONE

WASX7357I: By request, this scripting client is not connected to any
server process. Certain configuration and application operations
will be available in local mode.
WASX7029I: For help, enter: "$Help help"

wsadmin>securityoff
LOCAL OS security is off now but you need to restart server1 to make
it affected.

wsadmin>$AdminConfig save
wsadmin>exit
```


Messages

⚠ The security configuration is enabled or modified in a Network Deployment environment. The following steps need to be followed so that all the processes in this environment have the same security run-time settings: 1) Verify that all nodes are synchronized with these security configuration changes before stopping these processes. 2) If any node agents are currently stopped, issue a manual **syncNode** command before starting that node agent. 3) Stop all of the processes in the entire cell, including the deployment manager, node agents, and Application Servers. 4) Restart all of the processes in the cell; restart the deployment manager and node agents first, then Application Servers.

⚠ If the Restrict access to local resources option is not enabled, the Java virtual machine (JVM) system resources are not protected. For example, applications can read and write to files on file systems, listen to sockets, exit the Application Server process, and so on. However, by enabling the Restrict access to local resources option, applications might fail to run if the required permissions are not granted to the applications.

ℹ If any of the fields are changed, save the configuration and then stop and restart the server.

⚠ Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

An option to synchronize the configuration across multiple nodes after saving can be enabled in Preferences.

⚠ The server may need to be restarted for these changes to take effect.

Figure 4-8 Global security: Click Save and take note of additional messages and warnings

14. Save the configuration.

The stand-alone LDAP registry is configured at this point. The user realm will be given the system-defaulted realm name of the host concatenated to the port number separated by a colon (*host:port*). For this example, the realm will be `sys4.itso.ibm.com:389`.

4.2.4 Configuring a stand-alone LDAP using wsadmin commands

You can accomplish the same results by using the wsadmin and scripting environment.

The following example shows the commands that can be executed to configure a stand-alone LDAP registry. The commands use the jython scripting language and were executed using **`wsadmin`** in an interactive mode. (Original commands were captured courtesy of the administrative console Command Assistance).

Note: To keep the example simple, authentication settings, such as the authentication mechanism, SSO enablement, and so forth, are not handled in these scripts. The examples illustrate a minimal set of commands using authentication and authorization defaults. For example, the setAdminActiveSecuritySettings command can have many more options than shown here.

Follow these steps to configure a stand-alone LDAP registry using wsadmin commands:

1. Configure the LDAP search filters and object classes used for user and groups in the directory (Example 4-1).

Example 4-1 Configure LDAP search filters and object class mappings

```
AdminTask.configureAdminLDAPUserRegistry('[-userFilter
(&(uid=%v)(objectclass=inetOrgPerson)) -groupFilter
(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))
-userIdMap *:uid -groupIdMap *:cn -groupMemberIdMap
ibm-allGroups:member;ibm-allGroups:uniqueMember -certificateFilter
-certificateMapMode EXACT_DN -krbUserFilter
(&(krbPrincipalName=%v)(objectclass=inetOrgPerson)) -customProperties
["com.ibm.websphere.security.ldap.recursiveSearch="] -verifyRegistry
false ]')
```

2. Configure the general properties of the LDAP directory, including the host, port base DN, SSL configuration, and so forth (Example 4-2).

Example 4-2 Configure LDAP general properties

```
AdminTask.configureAdminLDAPUserRegistry('[-ldapHost
sys4.itso.ral.ibm.com -ldapPort 389 -ldapServerType
IBM_DIRECTORY_SERVER -baseDN ou=unit1,o=ibm,c=us -bindDN
uid=admin1,ou=adminusers,ou=unit1,o=ibm,c=us -bindPassword admin1pwd01
-searchTimeout 120 -reuseConnection true -sslEnabled false -sslConfig
-autoGenerateServerId true -primaryAdminId admin1 -ignoreCase true
-customProperties -verifyRegistry false ]')
```

3. Validate the LDAP configuration settings (Example 4-3).

Example 4-3 Validate LDAP configuration settings

```
AdminTask.configureAdminLDAPUserRegistry('[-verifyRegistry true ]')
```

4. Set the current registry type to be stand-alone LDAP and activate administrative security (Example 4-4 on page 83).

Example 4-4 Set current registry type

```
AdminTask.setAdminActiveSecuritySettings('[-activeUserRegistry
LDAPUserRegistry -enableGlobalSecurity true]')
```

5. Save the configuration changes (Example 4-5).

Example 4-5 Save configuration

```
AdminConfig.save()
```

Tip: While interactive mode is used for the wsadmin example here, the options being specified in the square brackets can equally be extracted to a properties file. A properties file model is more manageable for the administrator and can be integrated into the asset repository for each environment and directory configuration supported. An obvious security risk is shown here in that the bind password is not obfuscated. Be aware of this risk if you use scripting and do not store these passwords or scripts in an unprotected manner.

Optional LDAP failover configuration

When you configure a stand-alone LDAP for use in production systems, the availability of the user registry to facilitate authentication and provisioning of user information becomes vital to the availability of the applications hosted in the application server environment. Thus, it is very desirable to have a highly available configuration for LDAP.

When using a stand-alone LDAP, the failover list can be configured using wsadmin scripting. There is a sample wsadmin script provided in the Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_secfailover_ldap.html

This script adds additional LDAP directory instances to the existing WebSphere Application Server configuration. The added server and port configurations create an LDAP connection/failover list.

This script can be copied, saved, and executed for the purpose of configuring failover using WebSphere Application Server.

Figure 4-9 on page 84 shows how this environment can be extended to add sys1.itso.ral.ibm.com/389 as an additional LDAP in the user registry host list.

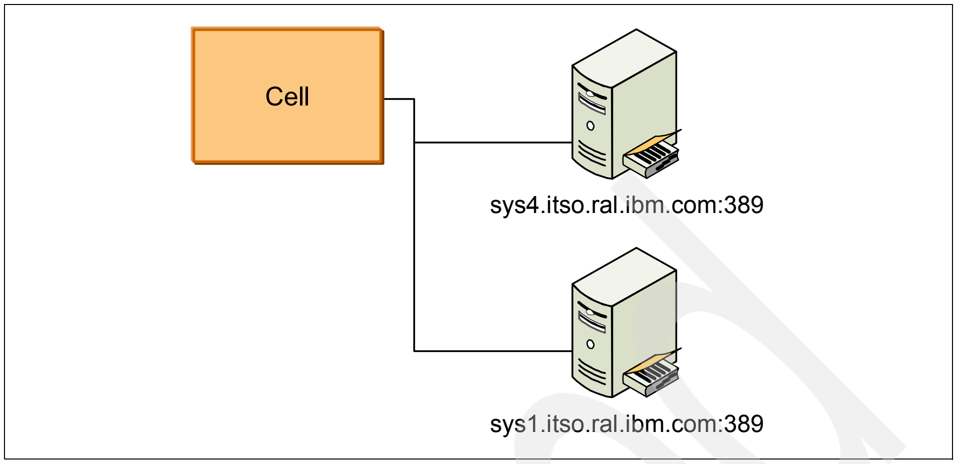


Figure 4-9 LDAP failover configuration illustration

Example 4-6 shows the command used to invoke the LDAPAdd.py script and the output that results from the script execution.

Example 4-6 LDAPAdd.py script execution example

```
C:\WebSphere\AppServer\profiles\LBDmgr03\bin>wsadmin -f
c:\scripts\LDAPAdd.py sys1.itso.ral.ibm.com 389
WASX7209I: Connected to process "dmgr" on node sys2CellManager03 using
SOAP connector;  The type of process is: DeploymentManager
WASX7303I: The following options are passed to the scripting
environment and are available as arguments that are stored in the argv
variable: "[sys1.itso.ral.ibm.com, 389]"
Got LDAPUserRegistry ConfigId is
(cells/sys2Cell03|security.xml#LDAPUserRegistry_1)

Got Security Mbean is
WebSphere:name=SecurityAdmin,process=dmgr,platform=proxy,node=sys2CellM
anager03,version=7.0.0.1,type=SecurityAdmin,mbeanIdentifier=SecurityAdm
in,cell=sys2Cell03,spec=1.0

Done setting up attributes values for LDAP User Registry
Updated was saved successfully


C:\WebSphere\AppServer\profiles\LBDmgr03\bin>
```

When binding to a stand-alone LDAP registry, the application server writes a log entry to the SystemOut.log. This log entry describes in detail to which LDAP instance it is connected, as shown in Example 4-7.

Example 4-7 LDAP connection log example

```
[2/25/09 17:27:23:875 EST] 00000012 LdapRegistryI A   SECJ0419I: The 
user registry is currently connected to the LDAP server 
ldap://sys4.itso.ral.ibm.com:389.
```

When this style of failover is configured, there are issues to consider:

- The list is not a primary/secondary configuration. Unlimited instances can be added to the list; however, preference is determined by the order of the list and is only considered when the security service is connecting to the LDAP directory:
  - Therefore, different servers in a cell can be bound to different LDAP directories. However, the stated expectation of this configuration is that the failover servers are the same in regard to schema and the content that they manage.
  - After an LDAP instance outage, servers will not fail back when the original server is again active.
- This list is also not a workload management mechanism. An application server instance connects to a server when it first binds to a directory and uses that directory until stopped or until the directory to which it is connected is determined to be unreachable.

4.2.5 Stand-alone LDAP dynamic and nested group configuration

Next, we describe the differences between stand-alone LDAP dynamic and nested group configuration.

Dynamic groups

A dynamic group defines its members differently than a static group. Instead of listing them individually, the dynamic group defines its members using an LDAP search. The dynamic group uses the structural objectclass groupOfURLs (or auxiliary objectclass ibm-dynamicGroup) and the attribute memberURL to define the search using a simplified ldap URL syntax:

`ldap:/// <base DN of search>? ?<scope of search>?<searchfilter>`

Note: From the ldap URL that is shown here, it is clear that the host name must not be present in the syntax. The remaining parameters are just like normal ldap URL syntax. Each parameter field must be separated by a question mark (?), even if no parameter is specified. Normally, a list of attributes to return is included between the base DN and the scope of the search. This parameter is also not used by the server when determining dynamic membership, and so, it can be omitted; however, the separator ? must still be present.

In the ldap URL syntax:

- Base DN of search

  This is the point from which the search begins in the directory. It can be the suffix or root of the directory, for example, `Austin`. This parameter is required.

- Scope of search

  This parameter specifies the extent of the search. The default scope is `base`:

  - `base` returns information only about the base DN specified in the URL.
  - `one` returns information about entries one level below the base DN specified in the URL. It does not include the base entry.
  - `sub` returns information about entries at all levels below and includes the base DN.

- Search filter

  This parameter is the filter that you want to apply to the entries within the scope of the search. The default is `objectclass=*`.

The search for dynamic members is always internal to the server, so unlike a full ldap URL, a host name and port number are never specified, and the protocol is always ldap (never "ldaps"). The memberURL attribute can contain any kind of URL, but the server only uses memberURLs beginning with `ldap:///` to determine dynamic membership.

For example, consider the following queries:

- A single entry in which the scope defaults to base and the filter defaults to objectclass=*:

  ```
  ldap:///cn=John Doe, cn=Employees, o=Acme, c=US
  ```

- All entries that are 1-level below cn=Employees and the filter defaults to objectclass=*:

  ```
  ldap:///cn=Employees, o=Acme, c=US??one
  ```

- All entries that are under o=Acme,c=us with the objectclass=person:

```
ldap:///o=Acme, c=US??sub?objectclass=person
```

Nested groups

The nesting of groups enables the creation of hierarchical relationships that can be used to define inherited group membership. A *nested group* is defined is a child group entry whose DN is referenced by an attribute contained within a parent group entry. A parent group is created by extending one of the structural group object classes (groupOfNames, groupOfUniqueNames, accessGroup, accessRole, or groupOfURLs) with the addition of the `ibm-nestedGroup` auxiliary object class. After nested group extension, zero or more `ibm-memberGroup` attributes can be added, with their values set to the DNs of nested child groups.

Figure 4-10 illustrates the concept of a nested group.

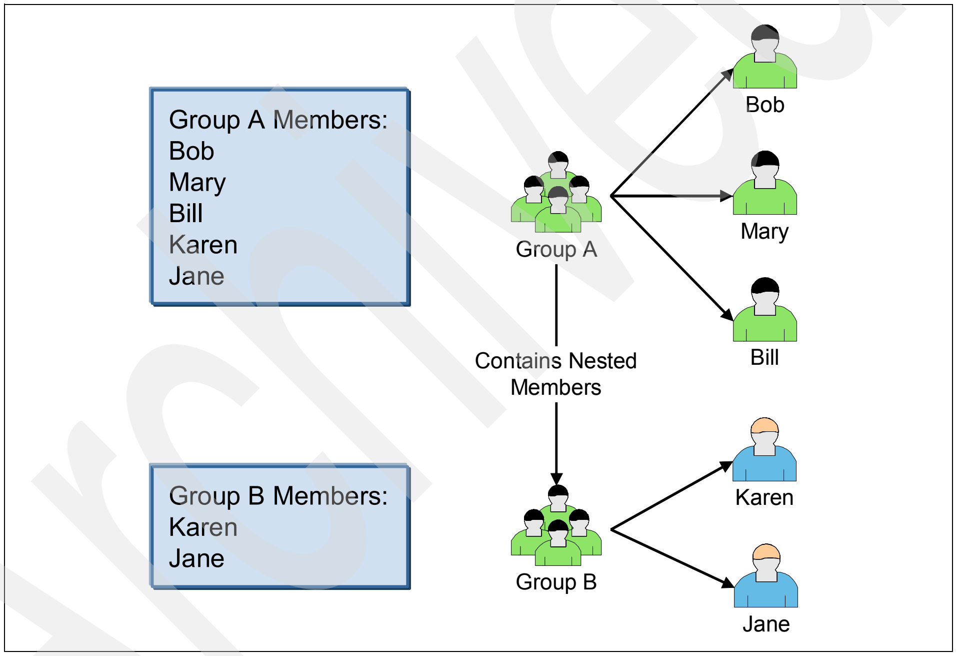


Figure 4-10 Nested group illustration

Configuring dynamic and nested groups for stand-alone LDAP

It is important to first note that dynamic and nested groups are not supported for all LDAP types. Be sure to check the Information Center or with IBM support to determine if the LDAP directory that you plan to use supports these configurations.

IBM Tivoli Directory Server supports both dynamic groups and nested groups. From the application server perspective, configuring the dynamic and nested group support is easy. But, how best to utilize dynamic and nested groups is a directory design consideration.

Next, we configure a dynamic group in WebSphere Application Server. The directory information tree is extended to include the dynamic group `dynAdminGroup` and the nested groups `inner` and `outer`. As shown in the directory information tree in Figure 4-11, note that the original static groups remain in the tree, but they are not shown in Figure 4-11.

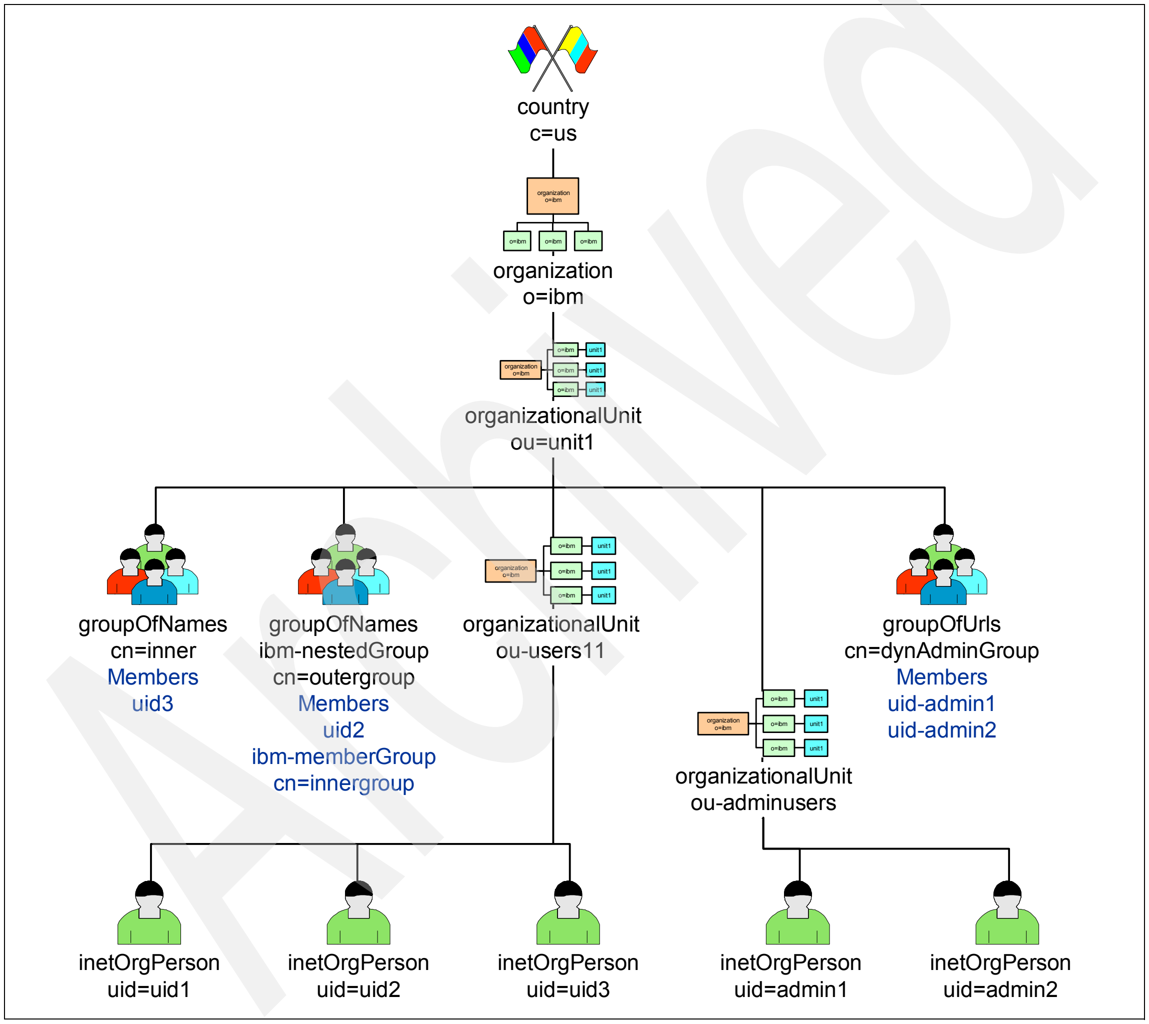


Figure 4-11 Directory information tree with dynamic and nested groups

When creating a dynamic group, the group specifies a memberURL attribute. The ldap query is placed in the memberURL. In Example 4-8 on page 89, we show an example query where a dynamic group called `dynAdminGroup` has been added to the LDAP directory with the member URL.

Example 4-8 Dynamic group added to directory

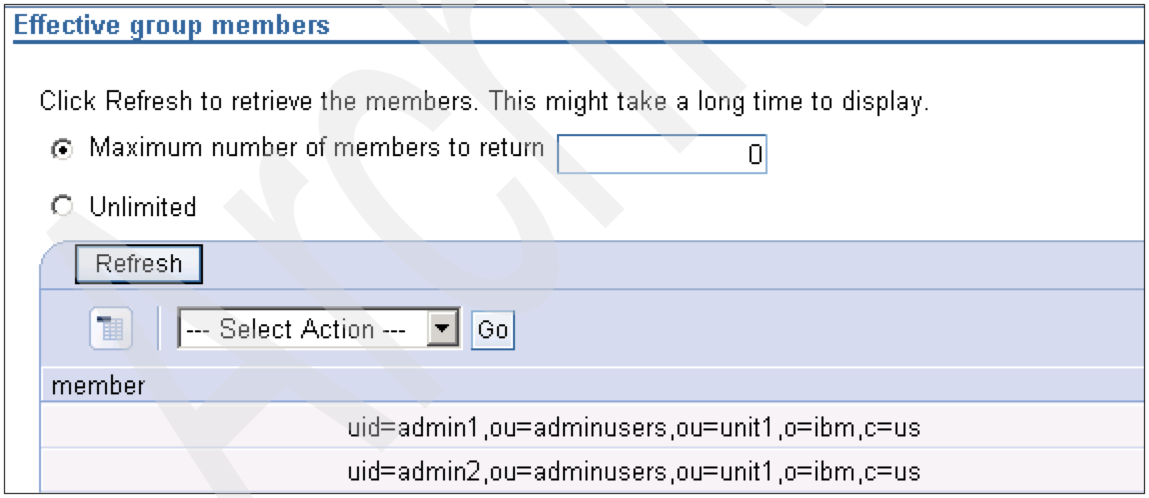
```
ldap:///ou=adminusers,ou=unit1,o=ibm,c=us??one?objectclass=inetOrgPerso
n
```

This LDAP query will include in the group all objects of class inetOrgPerson from the organizational unit adminusers. Whenever someone is added to this organizational unit or removed from it, the group is dynamically updated.

Important: The following screen captures are from the Web-based administration tool of IBM Tivoli Directory Server:

- Figure 4-12 on page 89
- Figure 4-13 on page 90
- Figure 4-14 on page 90
- Figure 4-15 on page 91

The effective group members can be found in the IBM Tivoli Directory Server, as shown in Figure 4-12.



Effective group members

Click Refresh to retrieve the members. This might take a long time to display.

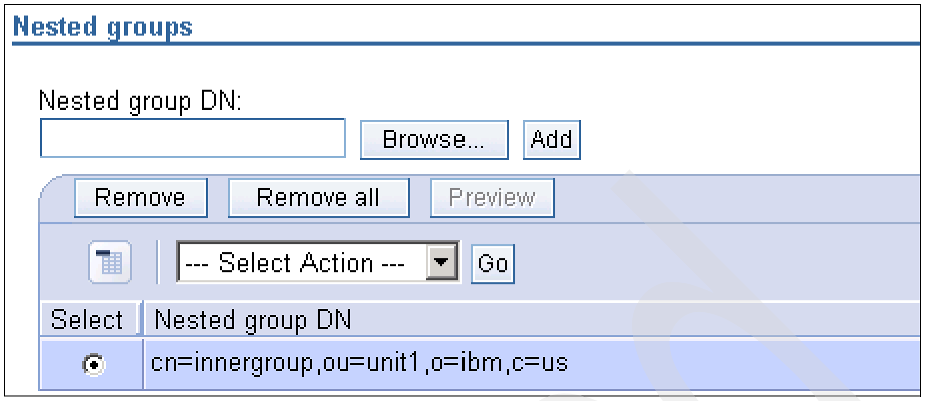
(●) Maximum number of members to return: 0

( ) Unlimited

member
uid=admin1,ou=adminusers,ou=unit1,o=ibm,c=us
uid=admin2,ou=adminusers,ou=unit1,o=ibm,c=us

Figure 4-12 Manage members: cn=dynAdminGroup,ou=unit1,o=ibm,c=us panel

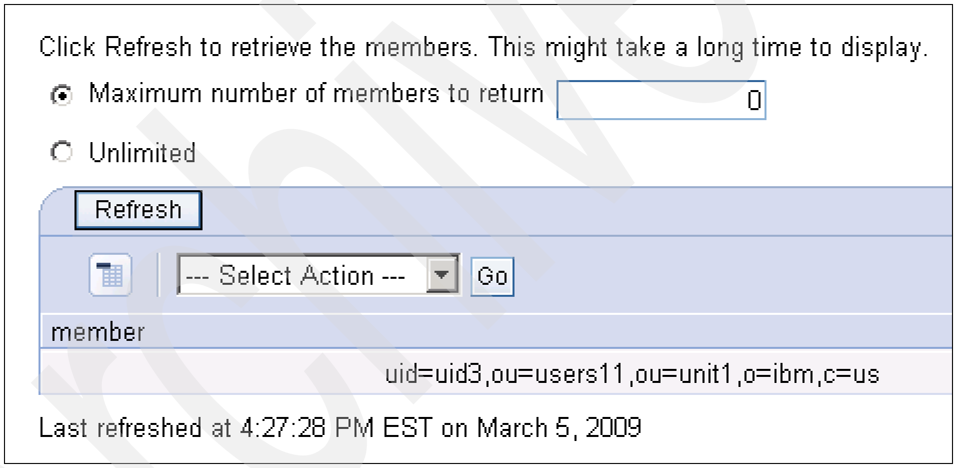
For the nested group scenario, a nested group called `innergroup` was created and then nested in a group called `outergroup` as shown in Figure 4-13 on page 90.



Select	Nested group DN
●	cn=innergroup,ou=unit1,o=ibm,c=us

Figure 4-13 Manage members: cn=outergroup,ou=unit1,o=ibm,c=us

The group members for `innergroup` are shown in Figure 4-14.


Click Refresh to retrieve the members. This might take a long time to display.

● Maximum number of members to return: 0

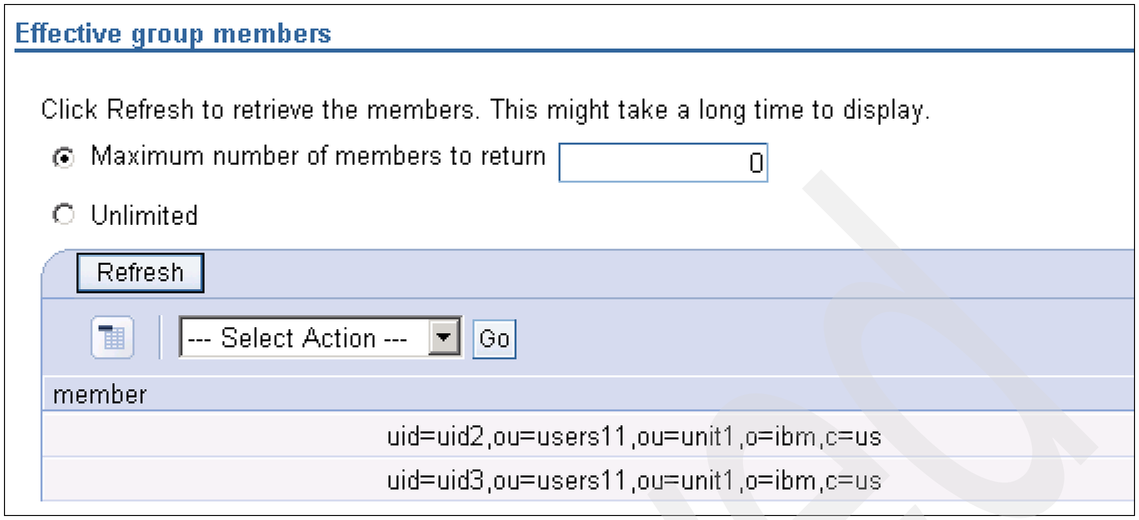
○ Unlimited

member
uid=uid3,ou=users11,ou=unit1,o=ibm,c=us

Last refreshed at 4:27:28 PM EST on March 5, 2009

Figure 4-14 Manage members: cn=innergroup,ou=unit1,o=ibm,c=us on Effective group members panel

Effective group members for the outer group are shown in Figure 4-15 on page 91 respectively.


Effective group members

Click Refresh to retrieve the members. This might take a long time to display.

[✓ Maximum number of members to return] 0

Unlimited

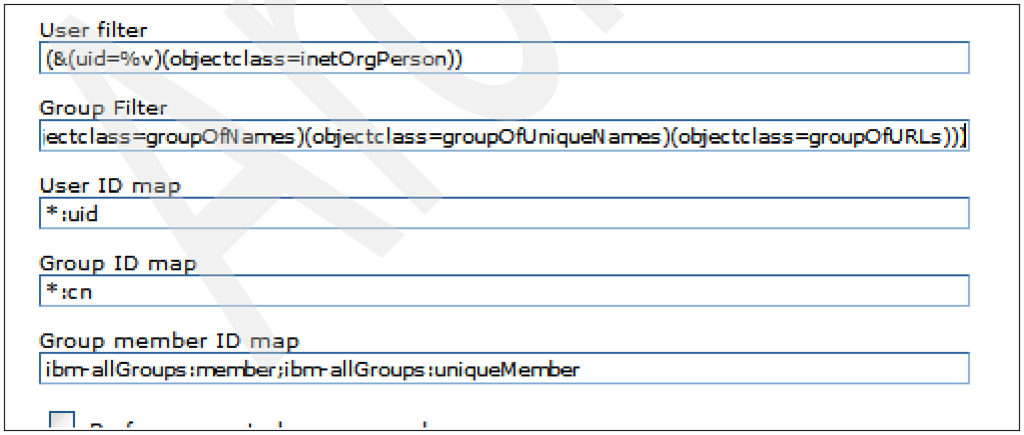
member
uid=uid2,ou=users11,ou=unit1,o=ibm,c=us
uid=uid3,ou=users11,ou=unit1,o=ibm,c=us

Figure 4-15 Manage members: cn=outergroup,ou=unit1,o=ibm,c=us in Effective group members panel

The next step is to update the LDAP settings in the WebSphere administrative console:

1. To configure dynamic and nested group support for the stand-alone LDAP configuration, modify the group filter and group ID settings.

   To find these settings, navigate to **Global security.** Select the **Standalone LDAP registry** realm, and click **Configure.** Then, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**. (Figure 4-16).


User filter
(&(uid=%v)(objectclass=inetOrgPerson))

Group Filter
jectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs))

User ID map
*:uid

Group ID map
*:cn

Group member ID map
ibm-allGroups:member;ibm-allGroups:uniqueMember

Figure 4-16 Add dynamic group object class groupOfURLs

2. In Figure 4-16 on page 91, make these changes:
   - Modify the Group Filter to include `(objectclass=groupOfURLs)`. The completed filter becomes:

     `(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))`

     (This complete filter cannot be shown in the screen capture shown in Figure 4-16 on page 91 due to its length)
   - Confirm that the Group member ID Map includes: ibm-allGroups:member;ibm-allGroups:uniqueMember

   Click **OK** to save the LDAP registry settings.
3. Click **OK** on the Standalone LDAP registry panel.
4. Click **Apply** on the Global security panel.
5. Save the configuration and restart the server environment.
6. (Optional) Verify that the group is available. If you are logged in as the primary administrative user, you can navigate to the Administrative Group roles panel from the Users and Groups menu ,and click **Add**.
7. (Optional) On the Administrative group roles → Group panel, use the search capability to show that the dynamic and nested groups are visible.

You have now completed the configuration for dynamic and nested groups in a stand-alone LDAP configuration using IBM Tivoli Directory Server.

4.2.6 Stand-alone LDAP configuration defaults

If you are considering using the security configuration wizard to configure your stand-alone LDAP but you are uncertain of the default object classes used, the following properties files in Example 4-9 show the default used for the various LDAP types.

Example 4-9 LDAP default configuration.properties

```
#
# This file contains the default filters used by WebSphere security for
# various LDAP Directory servers.
#
#
#ISMESSAGEFILE FALSE
#
-----------------------------------------------------------------------
--------------------------
```

```
types=netscape domino50 secureway actived iplanet ibm_dir_server
edirectory custom

#
------------------------------------------------------------------------
--------------------------

netscape.user.filter=(&(uid=%v)(objectclass=inetOrgPerson))
netscape.group.filter=(&(cn=%v)(|(objectclass=groupOfNames)(objectclass
=groupOfUniqueNames)))
netscape.user.idmap=inetOrgPerson:uid
netscape.group.idmap=*:cn
netscape.groupmember.idmap=groupOfNames:member;groupOfUniqueNames:uniqu
eMember
netscape.krbuser.filter=(&(hpKrbPrincipalName=%v)(objectcategory=inetOr
gPerson))

#
------------------------------------------------------------------------
--------------------------

domino50.user.filter=(&(uid=%v)(objectclass=Person))
domino50.group.filter=(&(cn=%v)(objectclass=dominoGroup))
domino50.user.idmap=person:uid
domino50.group.idmap=*:cn
domino50.groupmember.idmap=dominoGroup:member
# domino does not have a default attribute for Kerberos principal name.
domino50.krbuser.filter=(&(krbPrincipalName=%v)(objectcategory=Person))
#
------------------------------------------------------------------------
--------------------------

secureway.user.filter=(&(uid=%v)(objectclass=ePerson))
secureway.group.filter=(&(cn=%v)(|(objectclass=groupOfNames)(objectclas
s=groupOfUniqueNames)))
secureway.user.idmap=*:uid
secureway.group.idmap=*:cn
secureway.groupmember.idmap=groupOfNames:member;groupOfUniqueNames:uniq
ueMember
secureway.krbuser.filter=(&(krbPrincipalName=%v)(objectcategory=ePerson
))
#
------------------------------------------------------------------------
--------------------------
```

```
actived.user.filter=(&(sAMAccountName=%v)(objectcategory=user))
actived.group.filter=(&(cn=%v)(objectcategory=group))
actived.user.idmap=user:sAMAccountName
actived.group.idmap=*:cn
actived.groupmember.idmap=memberof:member
actived.krbuser.filter=(&(userprincipalname=%v)(objectcategory=user))
#
-----------------------------------------------------------------------
--------------------------

ibm_dir_server.user.filter=(&(uid=%v)(objectclass=ePerson))
ibm_dir_server.group.filter=(&(cn=%v)(|(objectclass=groupOfNames)(objec
tclass=groupOfUniqueNames)(objectclass=groupOfURLs)))
ibm_dir_server.user.idmap=*:uid
ibm_dir_server.group.idmap=*:cn
ibm_dir_server.groupmember.idmap=ibm-allGroups:member;ibm-allGroups:uni
queMember
ibm_dir_server.krbuser.filter=(&(krbPrincipalName=%v)(objectcategory=eP
erson))
#
-----------------------------------------------------------------------
--------------------------
iplanet.user.filter=(&(uid=%v)(objectclass=inetOrgPerson))
iplanet.group.filter=(&(cn=%v)(objectclass=ldapsubentry))
iplanet.user.idmap=inetOrgPerson:uid
iplanet.group.idmap=*:cn
iplanet.groupmember.idmap=nsRole:nsRole
iplanet.krbuser.filter=(&(krbPrincipalName=%v)(objectcategory=inetOrgPe
rson))

#
-----------------------------------------------------------------------
--------------------------

edirectory.user.filter=(&(cn=%v)(objectclass=Person))
edirectory.group.filter=(&(cn=%v)(objectclass=groupOfNames))
edirectory.user.idmap=person:cn
edirectory.group.idmap=*:cn
edirectory.groupmember.idmap=groupOfNames:member
edirectory.krbuser.filter=(&(krbPrincipalName=%v)(objectcategory=Person
))
```

4.3 Federated repositories

Organizations often grow in a fashion that does not always result in simple centralized repositories of user information. Also, applications often need to utilize information beyond the boundaries of their normal repository of information. Federated repositories acknowledge these requirements and provide a mechanism for unifying user information into a single view.

Federated repositories provide a unified view of the user information that is owned by multiple user repositories. Federated repositories support file-based, LDAP, database, and custom registry implementations. A federated repository can only be configured as the user registry at the cell security domain level, and only one federation is supported per cell.

Federated repositories are implemented in WebSphere Application Server by the Virtual Member Manager (VMM). Figure 4-17 on page 96 shows the functional components of the VMM.

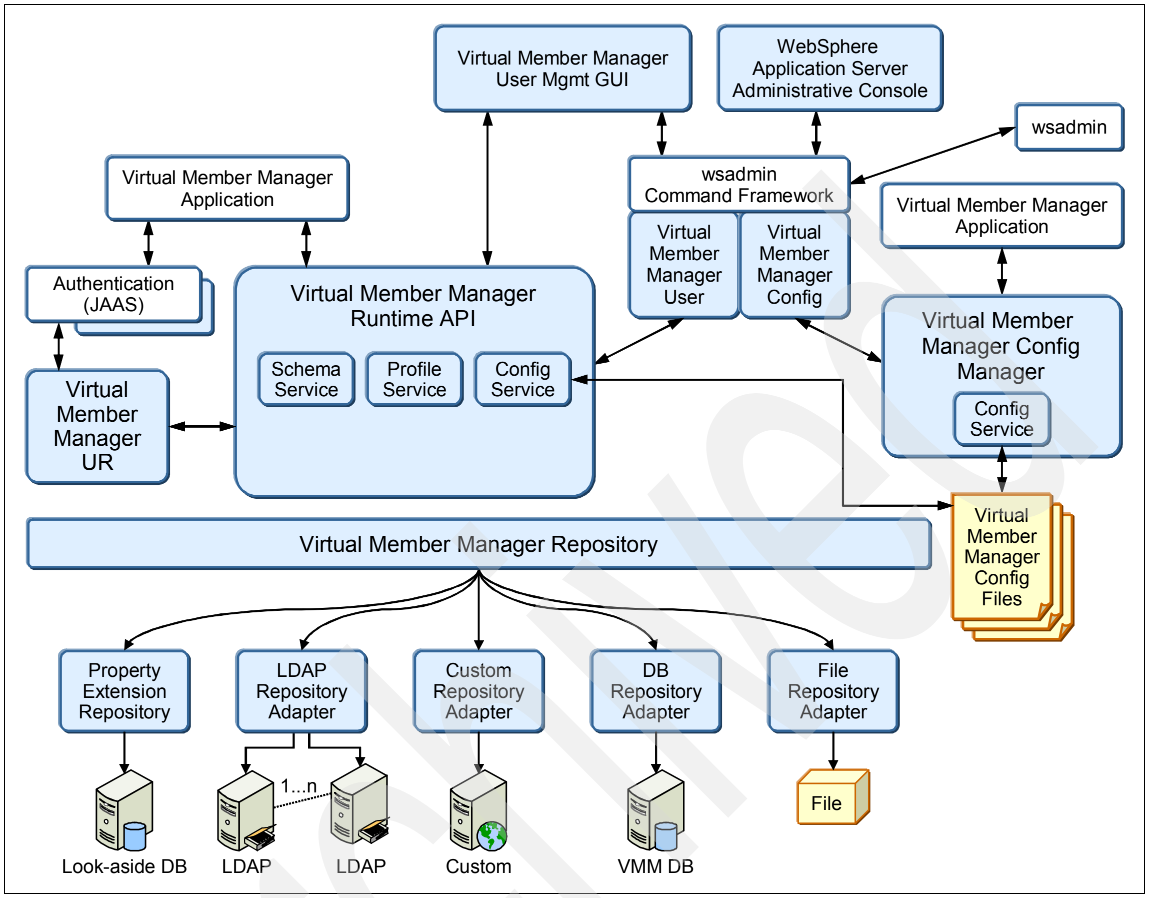

Virtual Member Manager User Mgmt GUI
WebSphere Application Server Administrative Console
wsadmin
wsadmin Command Framework
Virtual Member Manager Application
Virtual Member Manager User
Virtual Member Manager Config
Virtual Member Manager Application
Authentication (JAAS)
Virtual Member Manager Runtime API
Schema Service
Profile Service
Config Service
Virtual Member Manager Config Manager
Config Service
Virtual Member Manager UR
Virtual Member Manager Config Files
Virtual Member Manager Repository
Property Extension Repository
LDAP Repository Adapter
Custom Repository Adapter
DB Repository Adapter
File Repository Adapter
1...n
File
Look-aside DB
LDAP
LDAP
Custom
VMM DB

Figure 4-17 VMM functional component diagram

Beyond unifying user information into a single view, the VMM also has specialized components and features, which include but are not limited to:

- Along with normal search capabilities, VMM supports write access, such as create, update, and delete. This capability is provided for *one* of the federated repositories. This capability can be useful for managing administration-related groups.
- VMM has an adapter called the Property Extension Repository (PER) that permits the extensions of user attributes in a separate database. When retrieving user information, the VMM seamlessly joins the user information from the PER. The PER adapter permits the extension of existing repositories, for example, LDAP, that might not support certain attributes or that are configured as read-only. The VMM joins the attributes from the PER to the attributes of the extended repository as part of the repository queries.

- VMM has a component called the Entry Mapping Repository (EMR), which can be used to enhance performance when multiple respositories are federated. This component is used in two key scenarios:
    a. When multiple repositories are federated. If repositories are accessed using a unique identifier, the EMR can create a map to identify the owning repository for the unique identifier, thus saving the need to search all repositories.
    b. EMR is also used when one of the federated repositories does not have support for a unique identifier. EMR in this circumstance can be used to map a unique identifier and use this mapped identifier to identify the repository.
- When using a database repository, the database repository can be configured to contain group information about users from other repositories. The users do not need to be in the database repository. This style of configuration is a feature of the VMM and the database repository that is not supported in the file repository or the LDAP directories.
- VMM has a programming API that can be used for applications to interface with the repositories for user information.

Note: In this chapter, we will discuss elements of features that focus on the configuration of these features. But we do not discuss custom features, such as programming to the API or implementing a custom registry using the VMM SPI. For more information about custom interactions, refer to the Information Centers at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rwim_dev_vmmca.html`

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.wim.doc.en/repositoryspi.html`

There are important design considerations that can impact whether to use federated repositories in a particular environment. A short summary of well-known considerations includes:

- There are restrictions concerning repository type combinations and how many separate repository types can be federated together. Refer to:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.wim.doc.en/supportedconfigs.html`
- User identities must be unique across all federated repositories.
- There are restrictions about naming realms and how to name the federated repository when the cell is integrated with single sign-on (SSO) and

LDAP-aware products, such as IBM Tivoli Access Manager. Refer to "Understanding user realms when using federated repositories" on page 100 for more information.

- The federated repository is globally scoped and shared by all domains and applications that use the global security scope.
- The federated repositories function is a relatively new function in WebSphere Application Server. Configurations that include mixed cells incorporating older versions of WebSphere Application Server V6.0.*x* and earlier do not support federated repositories easily. We do not recommend using federated repositories as a configuration in a mixed cell environment.

4.3.1 Configuration checklist

The checklist in Table 4-2 is an abbreviated set of steps that can be used in planning the configuration of the federated repositories.

Table 4-2 Configuration checklist

Check item	Description	Reference

1	Verify the chosen repositories are in the compatible software list and that can be configured in the planned combination.	`http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369`
		
		`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.wim.doc.en/supportedconfigs.html`

2	Configure repositories to communicate using SSL where appropriate, such as LDAP repositories.	5.4, "Advanced concepts" on page 184

Check item	Description	Reference
3	Identify the general property configurations that are needed, which, for LDAP repositories, include: - Host - Port - Bind DN - Bind password - LDAP entity types - Group membership attribute	4.2.2, “Understanding the directory structure” on page 71 Figure 4-20 on page 106
4	Identify object class filters for user and group filters.	4.2.2, “Understanding the directory structure” on page 71 Figure 4-5 on page 76 Figure 4-6 on page 77
5	Ensure that user identifiers are unique across all federated repositories.	N/A
6	Identify whether support adapters, such as EMR and PER, are required.	4.3.5, “Configuring VMM database base adapter features” on page 121
7	Determine which of the federated repositories will be the base repository that supports update accesses from the cell. Note that there are requirements depending on the repository combinations.	`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.wim.doc.en/supportedconfigs.html`
8	Exchange SSL certificates with LDAP.	5.4, “Advanced concepts” on page 184
9	Choose authentication and authorization mechanisms, which can include: - Configure SSO - Require SSL for RMI/IIOP - Identify any realms that might need to be trusted	4.4, “Authentication and authorization settings” on page 133

4.3.2 Understanding user realms when using federated repositories

In a stand-alone security configuration, the realm maps one-to-one with the user registry, which is not true for federated repositories. The VMM extends the individual repositories into its own realm forming an umbrella realm of the federated repositories.

Each repository that is federated maps to a base entry in the federated realm, which further maps to the base entry in the actual repository. This mapping joins the repository to the realm of the federated repository, creating a single logical namespace.

In other words, a *realm* is a collection of independent repositories where each repository has:

- A repository name, which has an arbitrary value
- A base entry in the realm, which is a logical root entry for this particular repository within the federated repository (*virtual realm*)

For example, for an LDAP, the realm is the base DN in LDAP (from where searches start in the directory tree):

- Base entry in realm: `o=ldap,o=vmm`
- Base entry in LDAP repository: `ou=unit1,o=ibm,c=us`
- If the user in LDAP has a DN of `uid=adminuser,ou=adminusers,ou=unit1,o=ibm,c=us`, the DN in VMM is then `uid=adminuser,ou=adminusers,o=ldap,o=vmm`.

A consideration for configuring VMM to integrate with LDAP-related products for SSO and other similar features arises in relation to the user realm. Many of the products being integrated might only support one LDAP. For the trust association and SSO to work transparently, the distinguished names of the LDAP and the VMM realms must match. For example, when IBM Lotus® Domino® or WebSEAL sends a DN to WebSphere Application Server, it has to be the same DN that VMM uses. For further information about configuring VMM to work with this style of integration, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rwim_limitations.html

4.3.3 VMM entity types

VMM has built-in a set of entity types that are used to map the various entity types of the federated repositories. The federated repository entity types cannot be modified. They represent the groups, containers, and users in the federated repositories, and when each repository is federated, the objects from the federated repository are mapped to these entity types.

The default entity types are:

- Group
- OrgContainer
- PersonAccount

4.3.4 Configuring an LDAP federated repository using the console

File repositories and LDAP directories are the only federated repository types that are supported through the administrative console. Other repository types are only supported from **`wsadmin`** at this time.

Tip: The file repository was introduced to promote security being made active at the time of profile creation. But, it was never intended to be the primary user registry for a production environment, and it not suitable for large numbers of entries. While file repositories can be used as part of the federated repository, the preference for production configuration is to use highly available LDAP configurations.

Important: This configuration example presents the steps to configure an LDAP directory. In this example, the LDAP directory is an IBM Tivoli Directory Server instance.

It is important that you understand the LDAP configuration that you will use. Refer to 4.2.2, "Understanding the directory structure" on page 71, which highlights points that are also relevant in the preparation of a federated LDAP configuration.

VMM for WebSphere Application Server uses separate object class defaults than the object class defaults that are used for stand-alone configurations. If you are familiar with stand-alone configurations, you might already know that the default for IBM Tivoli Directory Server is `ePerson` for the user object class. While for the group, `groupOfNames` and `groupOfUniqueNames` are the default object classes.

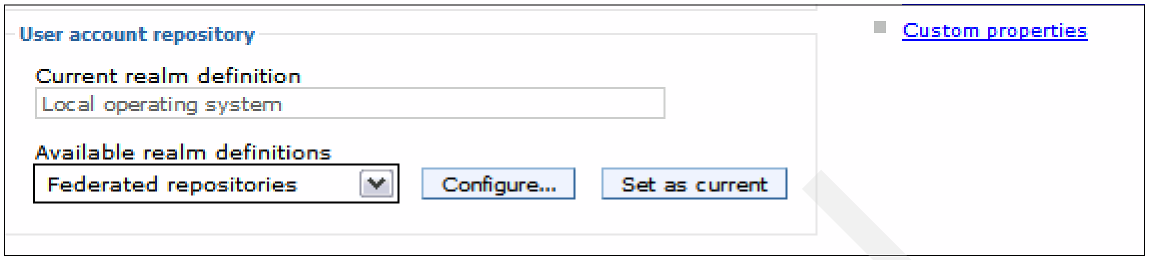
The defaults for VMM with regard to IBM Tivoli Directory Server are `inetOrgPerson` for the user object class and `groupOfNames` for the group object class. You can modify these user object class and group object class defaults after the initial configuration.

For the bind user test to work, a user of the default object class must be used. The defaults can be modified by changing the `wimconfig.xml` file. For more information about the default object classes that are used by VMM from various directory types, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.wim.doc.en/defaultldapconfigurationmappingbasedonldapservertype.html

To configure a federated repository that has an LDAP directory, perform the following steps:

1. Open the administrative console, and select **Security** → **Global security** (Figure 4-18 on page 103).


User account repository

Current realm definition

Local operating system

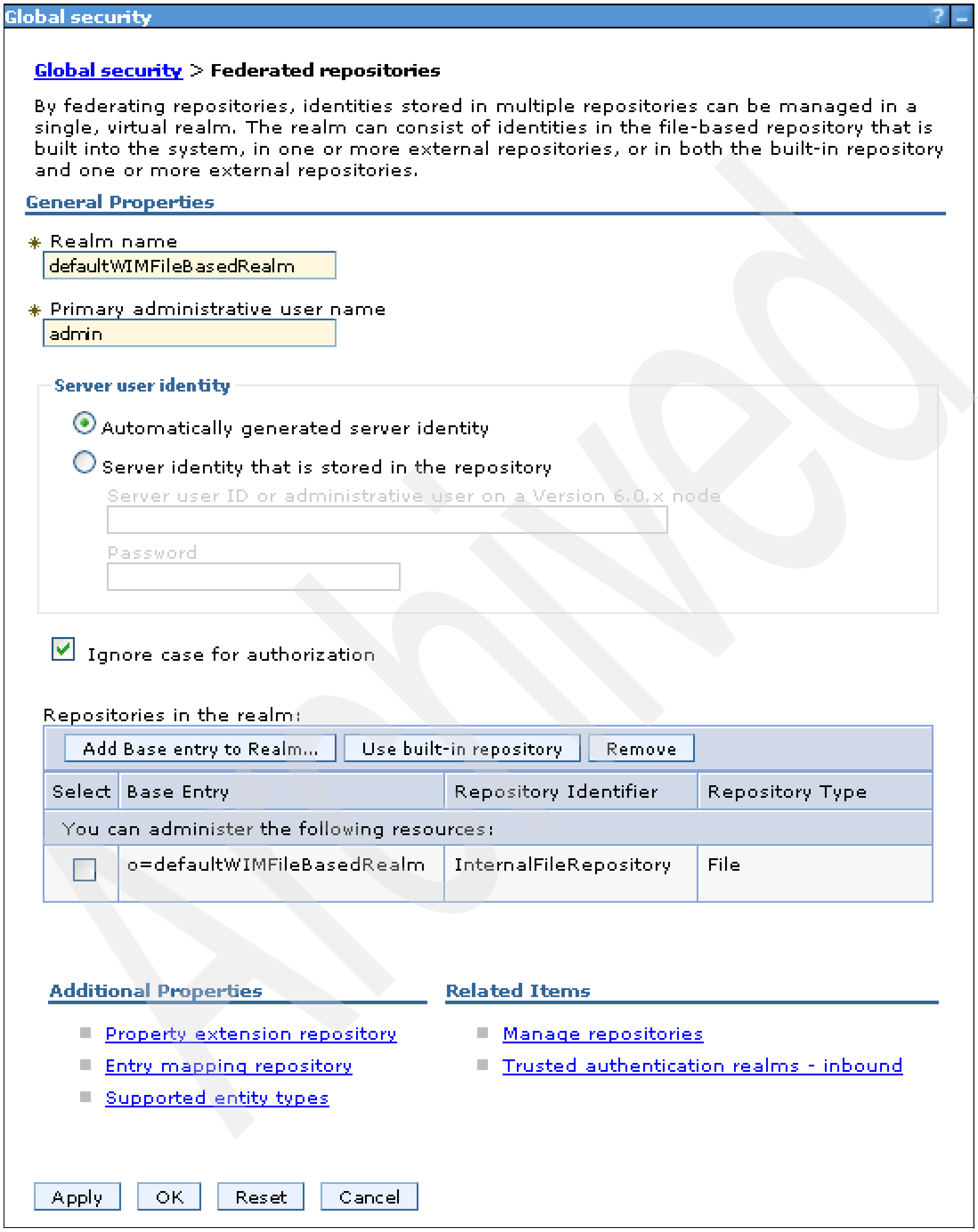
Available realm definitions

Federated repositories

Figure 4-18 Global security

2. In the User account repository section of the panel, select **Federated Repositories** from the Available realm definitions drop-down list (Figure 4-18).
3. Click **Configure** to open the Federated repositories configuration page (Figure 4-19 on page 104). Many smaller configuration tasks that collaborate to complete the configuration of a federated repository are initiated from this page.

Note: On the first navigation to the Federated repositories configuration page, a default realm name and file repository are created. This realm name and file repository also exist if security was activated at profile creation, with the addition that a primary administration ID is defined based on the user ID and password passed as input during profile creation.



Global security

Global security > Federated repositories

By federating repositories, identities stored in multiple repositories can be managed in a single, virtual realm. The realm can consist of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in repository and one or more external repositories.

General Properties

* Realm name
defaultWIMFileBasedRealm

* Primary administrative user name
admin

Server user identity

[✓ Automatically generated server identity]
Server identity that is stored in the repository
Server user ID or administrative user on a Version 6.0.x node
Password

[✓] Ignore case for authorization

Repositories in the realm:

Add Base entry to Realm... | Use built-in repository | Remove

Select	Base Entry	Repository Identifier	Repository Type
You can administer the following resources:			
☐	o=defaultWIMFileBasedRealm	InternalFileRepository	File

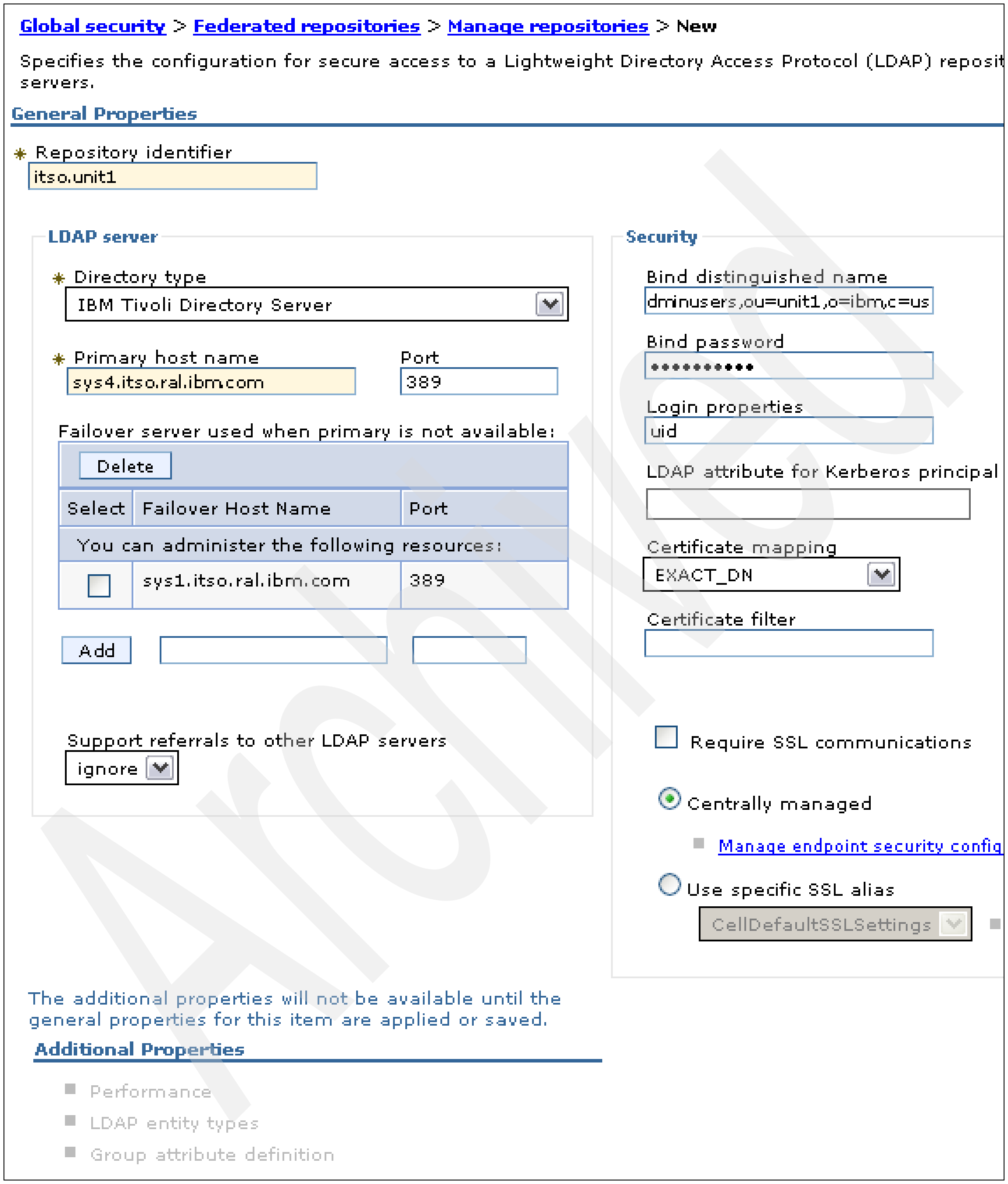
Additional Properties
- Property extension repository
- Entry mapping repository
- Supported entity types

Related Items
- Manage repositories
- Trusted authentication realms - inbound

Apply | OK | Reset | Cancel

Figure 4-19 Federated repository list

4. To proceed with the configuration of the LDAP directory as the base repository, click **Manage repositories** in the Related Items section of the panel (Figure 4-19 on page 104).
5. Click **Add** in the Manage repositories panel.
6. Provide the General properties information as shown in Figure 4-20 on page 106.


Global security > Federated repositories > Manage repositories > New

Specifies the configuration for secure access to a Lightweight Directory Access Protocol (LDAP) reposit
servers.

General Properties

* Repository identifier: itso.unit1

LDAP server

* Directory type: IBM Tivoli Directory Server

* Primary host name: sys4.itso.ral.ibm.com
Port: 389

Failover server used when primary is not available:

Delete

Select	Failover Host Name	Port
You can administer the following resources:		
☐	sys1.itso.ral.ibm.com	389

Add

Support referrals to other LDAP servers: ignore

Security

Bind distinguished name: dminusers,ou=unit1,o=ibm,c=us

Bind password: ••••••••••

Login properties: uid

LDAP attribute for Kerberos principal:

Certificate mapping: EXACT_DN

Certificate filter:

☐ Require SSL communications

● Centrally managed

Manage endpoint security config

○ Use specific SSL alias: CellDefaultSSLSettings

The additional properties will not be available until the general properties for this item are applied or saved.

Additional Properties

- Performance
- LDAP entity types
- Group attribute definition

Figure 4-20 Manage new repositories: LDAP

7. The minimal General properties information includes:
   - Repository Identifier: `itso.unit1`
   - Primary host name: `sys4.itso.ral.ibm.com`
   - Port (for primary host name): `389`
   - (Optional) Host name and port for failover servers:

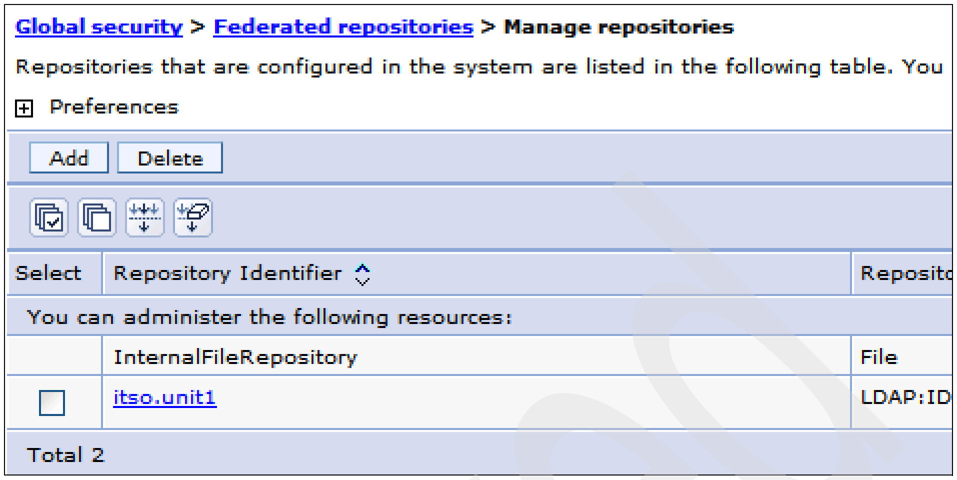
     `sys1.itso.ral.ibm.com 389`
   - Bind Distinguished Name

     `uid=admin1,ou=adminusers,ou=unit1,o=ibm,c=us`
   - Bind password: `adminpwd01`
   - Login properties (the default is `uid`)
   - (Optional but recommended) Select **Require SSL communications**

   Tip: Remember when adding failover servers to the configuration:
   - ► The bind test will test a bind with failover servers, as well as primary servers, so all servers must be running at configuration time.
   - ► When in failover mode, the cell will attempt to fail back to the primary server every 15 minutes.
   - ► You must complete certificate exchange for SSL communications for the failover LDAP directories, as well as the primary.

   Note that the Additional Properties cannot be completed at this time.

   Click **OK**.
8. Click **Save**, and click the link to the newly created repository `itso.unit1` (Figure 4-21 on page 108).

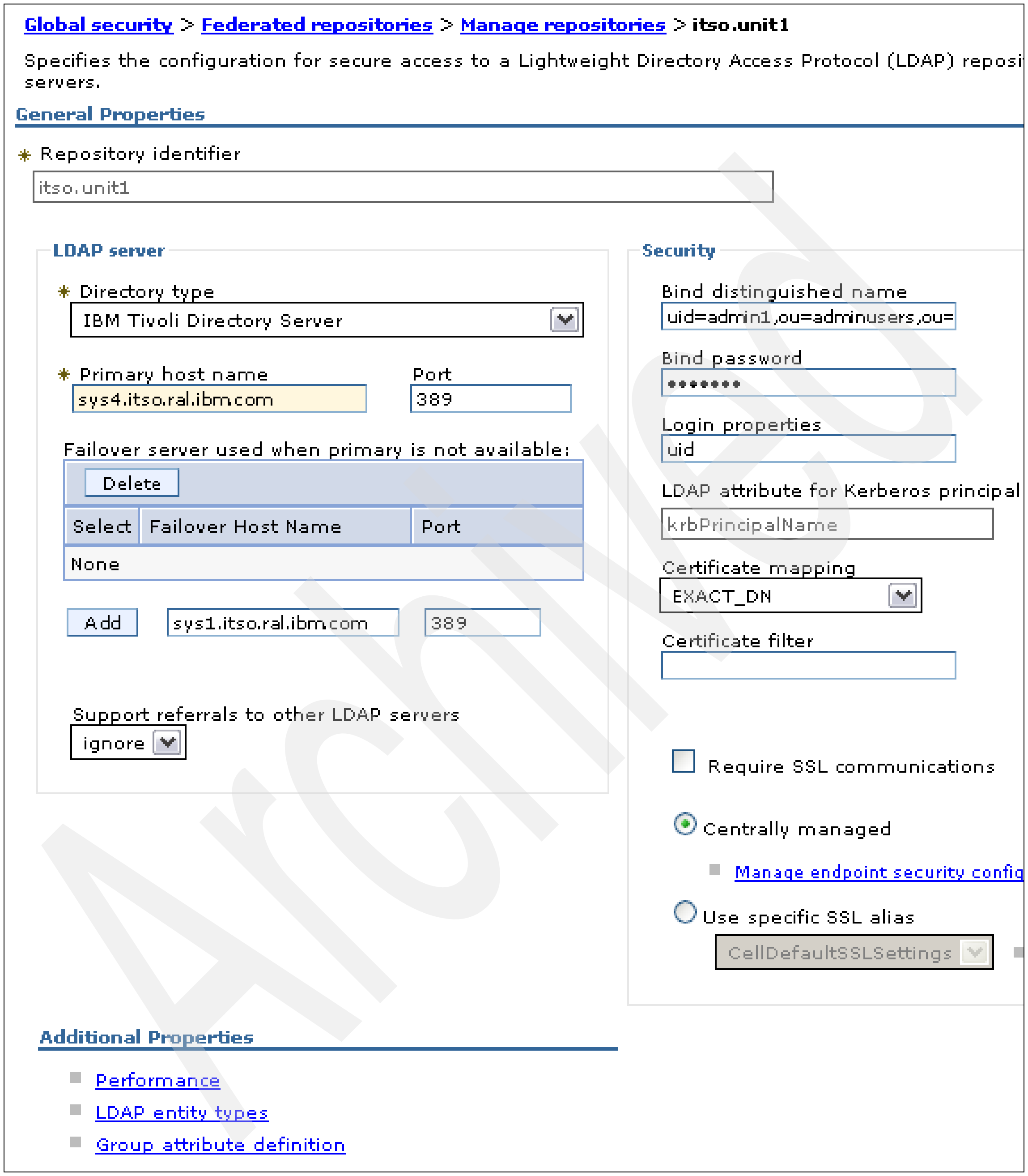

Global security > Federated repositories > Manage repositories

Repositories that are configured in the system are listed in the following table. You

Select	Repository Identifier	Reposito
You can administer the following resources:		
	InternalFileRepository	File
☐	itso.unit1	LDAP:ID
Total 2		

Figure 4-21 Manage repositories panel: Select the new repository

9. The links in the Additional Properties section are now active and the repository identifier is now read-only (Figure 4-22 on page 109).


Global security > Federated repositories > Manage repositories > itso.unit1

Specifies the configuration for secure access to a Lightweight Directory Access Protocol (LDAP) reposi
servers.

General Properties

* Repository identifier
itso.unit1

LDAP server

* Directory type
IBM Tivoli Directory Server

* Primary host name: sys4.itso.ral.ibm.com
Port: 389

Failover server used when primary is not available:

Select	Failover Host Name	Port
None		

Add: sys1.itso.ral.ibm.com 389

Support referrals to other LDAP servers
ignore

Security

Bind distinguished name
uid=admin1,ou=adminusers,ou=

Bind password
•••••••

Login properties
uid

LDAP attribute for Kerberos principal
krbPrincipalName

Certificate mapping
EXACT_DN

Certificate filter

☐ Require SSL communications

[✓ Centrally managed]

Manage endpoint security config

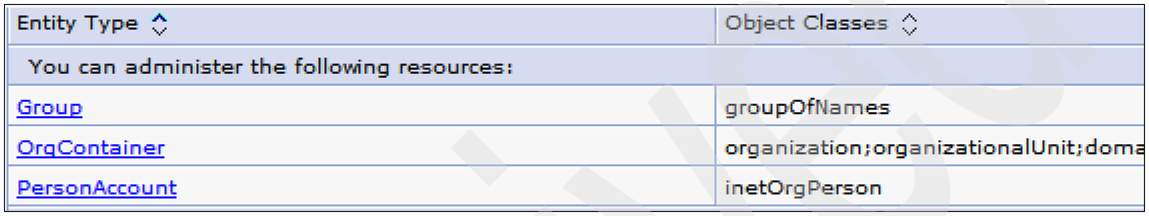
Use specific SSL alias
CellDefaultSSLSettings

Additional Properties

- Performance
- LDAP entity types
- Group attribute definition

Figure 4-22 The itso.unit1 configuration panel

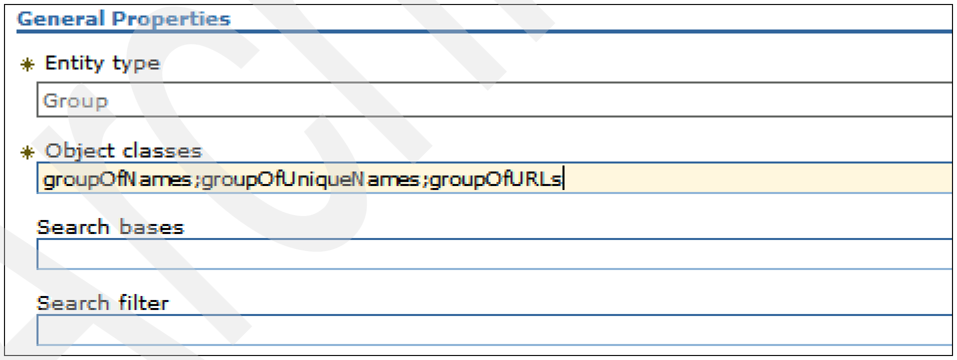
10. We mentioned in the beginning of this configuration process that the default object class that is supported for groups with IBM Tivoli Directory Server is `groupOfNames`. In the directory information tree for the directory being configured here (Figure 4-2 on page 72), we use the object class `groupOfUniqueNames`, also. During the stand-alone configuration example ("Stand-alone LDAP dynamic and nested group configuration" on page 85), a dynamic group of type `groupOfURLs` was also added. For the cell to understand these group types, the object class must be added to the repository's list of group object classes.
11. To modify the group classes, select the **LDAP entity types**.
12. Select **Group** (Figure 4-23).



Entity Type	Object Classes
You can administer the following resources:	
Group	groupOfNames
OrgContainer	organization;organizationalUnit;doma
PersonAccount	inetOrgPerson

Figure 4-23 Group entity type

13. In the General properties section, add the additional object classes of `groupOfUniqueNames` and `groupOfURLs` (Figure 4-24).



General Properties

* Entity type
Group

* Object classes
groupOfNames;groupOfUniqueNames;groupOfURLs

Search bases

Search filter

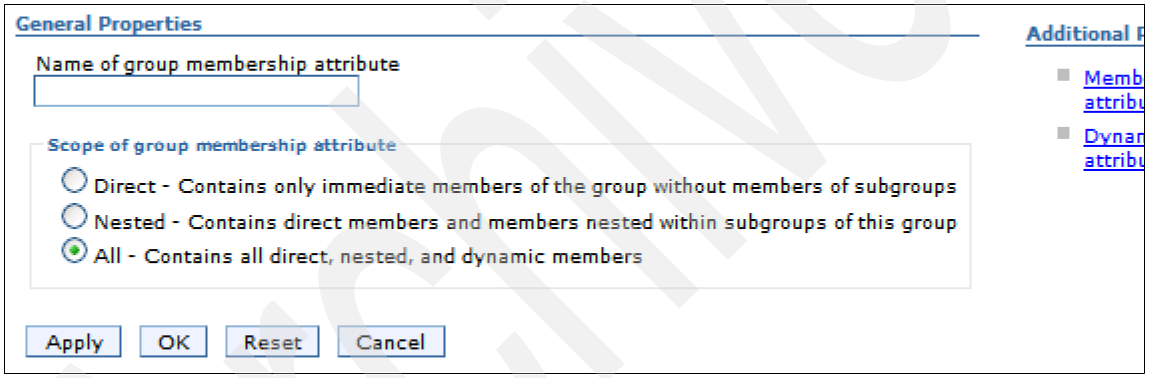
Figure 4-24 Add additional object classes

Note: It is valuable to highlight two other items here:

- On the panel that is shown in Figure 4-24, you can specify particular search bases for the entity type.
- You can also explicitly set search filters on the panel that is shown in Figure 4-24. If a search filter is not specified, the object classes and the relative distinguished name (RDN®) properties are used to generate the search filter.

Click **OK.**

14. Use the navigation trail at the top of the page to return to the Manage repositories page. Click **itso.unit1** to open the configuration page (Figure 4-22 on page 109).
15. Select **Group attribute definition** from the Additional Properties section of the panel (Figure 4-22 on page 109).


General Properties

Name of group membership attribute

Scope of group membership attribute

( ) Direct - Contains only immediate members of the group without members of subgroups

( ) Nested - Contains direct members and members nested within subgroups of this group

[✓ All - Contains all direct, nested, and dynamic members]

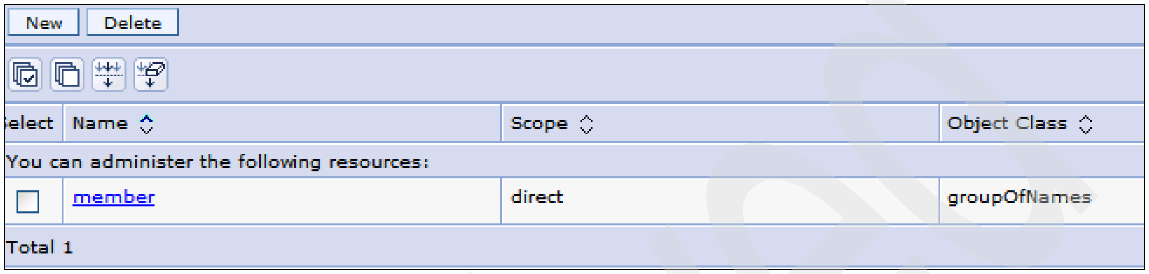
Figure 4-25 Group attribute definition panel

Most LDAP directories have a standard attribute on every person object that identifies the groups of which the object is a member. This attribute varies from directory to directory. Certain directories do not support this attribute, in which case, the application server needs to perform a search of all of the group types to identify the user group relationship. For IBM Tivoli Directory Server, the attribute is `ibm-allGroups`, which will be set in the “Name of group membership attribute” field in subsequent steps.

The radio selection on the panel in Figure 4-25 also identifies the types of mapping that are supported by the attribute. Again, the type of mapping that is supported can vary depending on the directory server that is being used.

But before completing the panel that is shown in Figure 4-25 on page 111, you must first configure the Additional properties. Configure the static or direct groups by selecting **Member attributes**, and set the dynamic groups by using **Dynamic member attributes**.

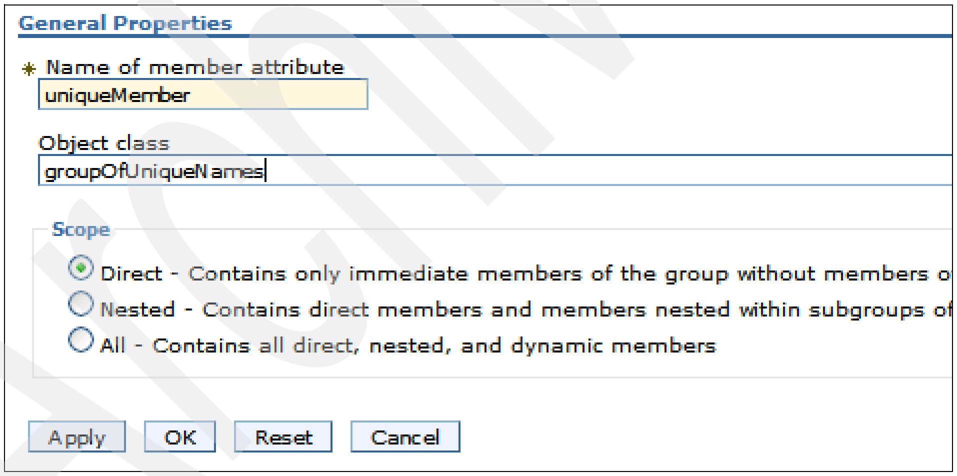
16. In Figure 4-25 on page 111, select **Member attributes** in the Additional Properties section to open the next panel (Figure 4-26). Note the default group, `groupOfNames`, is already mapped with the attribute "`member`."



Select	Name	Scope	Object Class
You can administer the following resources:			
☐	member	direct	groupOfNames
Total 1			

Figure 4-26 List of member attributes

17. Click **New**. Figure 4-27 is displayed.



* Name of member attribute: uniqueMember

Object class: groupOfUniqueNames

Scope

(●) Direct - Contains only immediate members of the group without members o
( ) Nested - Contains direct members and members nested within subgroups of
( ) All - Contains all direct, nested, and dynamic members

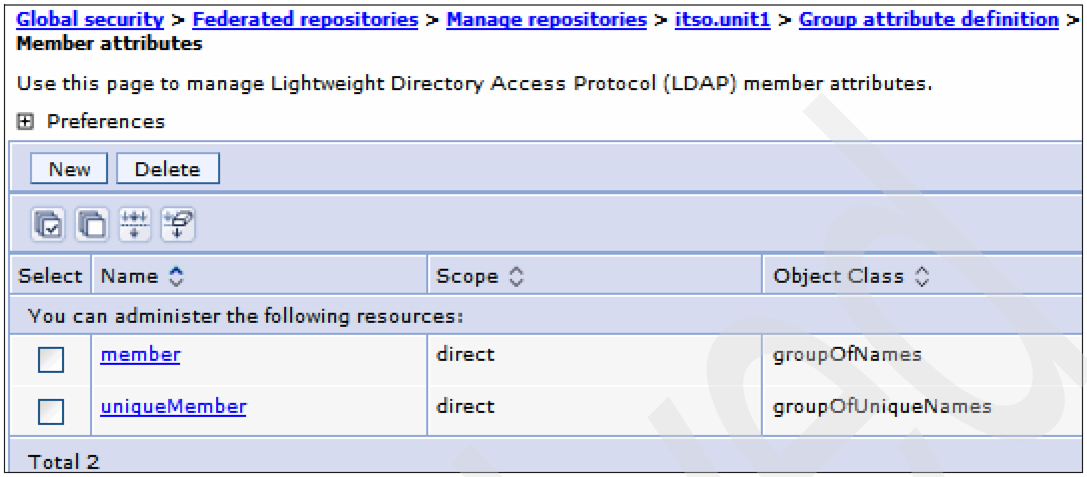
Figure 4-27 Define a new member attribute

18. Enter the values for the `groupOfUniqueNames` (Figure 4-27):

 – Name of member attribute: `uniqueMember`
 – Object class: `groupOfUniqueNames`
 – Scope: Select **Direct**

 (groupOfUniqueNames is a static list that is directly referenced)

Click **OK.** You see the new attribute definition in the list of member attributes (Figure 4-28).

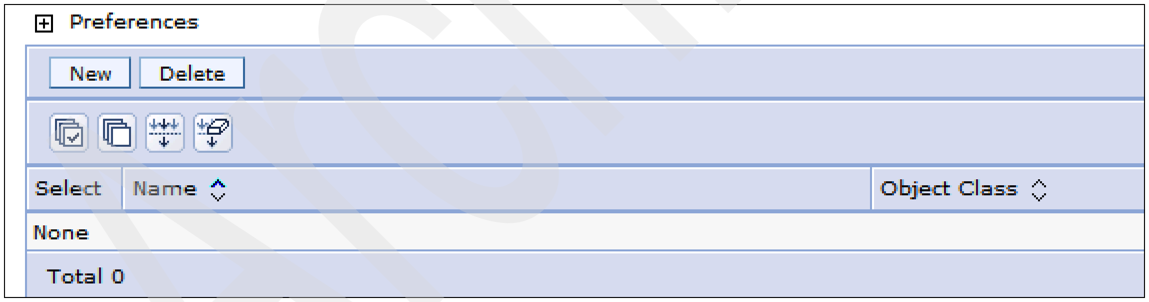

Global security > Federated repositories > Manage repositories > itso.unit1 > Group attribute definition > Member attributes

Use this page to manage Lightweight Directory Access Protocol (LDAP) member attributes.

Select	Name	Scope	Object Class
You can administer the following resources:			
☐	member	direct	groupOfNames
☐	uniqueMember	direct	groupOfUniqueNames
Total 2			

Figure 4-28 Member attribute panel

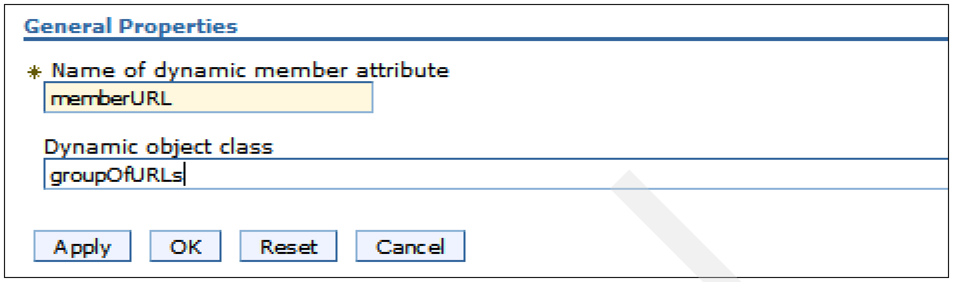
19. Click the **Group attribute definition** navigation link in the navigation trail to go back to the Group attribute definition panel (Figure 4-25 on page 111), and select **Dynamic member attributes** in the Additional Properties section.
20. Click **New** (Figure 4-29).



Select	Name	Object Class
None		
Total 0		

Figure 4-29 Add a new dynamic member attribute

21. Enter the values for the fields as shown in Figure 4-30 on page 114.



General Properties

* Name of dynamic member attribute
memberURL

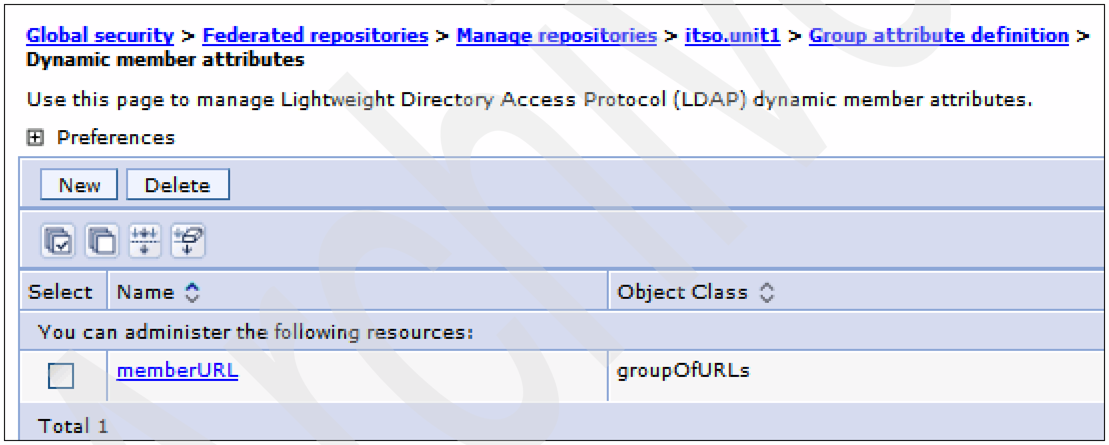
Dynamic object class
groupOfURLs

Figure 4-30 New dynamic attribute values

Enter:

– Name of dynamic member attribute: `memberURL`
– Object Class: `groupOfURLs`

Click **OK**. You will see the new attribute listed in the Dynamic member attributes panel (Figure 4-31).



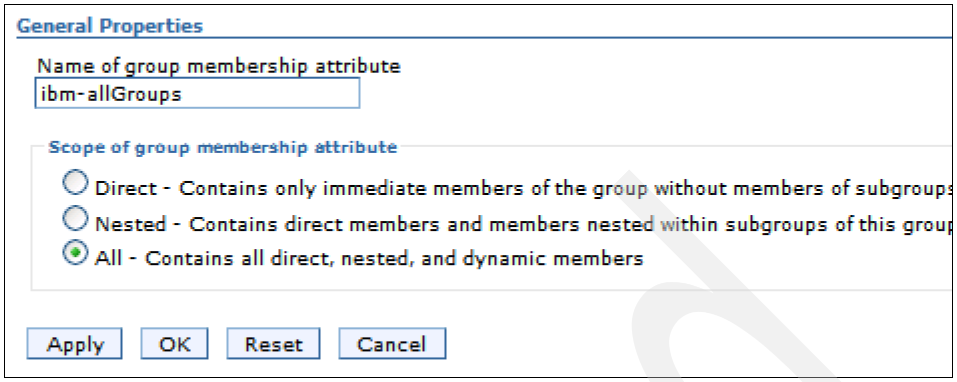
Global security > Federated repositories > Manage repositories > itso.unit1 > Group attribute definition > **Dynamic member attributes**

Use this page to manage Lightweight Directory Access Protocol (LDAP) dynamic member attributes.

Select	Name	Object Class
You can administer the following resources:		
☐	memberURL	groupOfURLs
Total 1		

Figure 4-31 New dynamic attribute

22. Click the **Group attribute definition** link in the navigation trail in Figure 4-31 to return to the Group attribute definition panel. Complete the panel as shown in Figure 4-32 on page 115.


General Properties

Name of group membership attribute
ibm-allGroups

Scope of group membership attribute

( ) Direct - Contains only immediate members of the group without members of subgroups
( ) Nested - Contains direct members and members nested within subgroups of this group
(●) All - Contains all direct, nested, and dynamic members

Apply | OK | Reset | Cancel

Figure 4-32 Complete the group attribute definition

Enter:

– Name of group membership attribute: `ibm-allGroups`
– Scope of group membership attribute: Select **All**

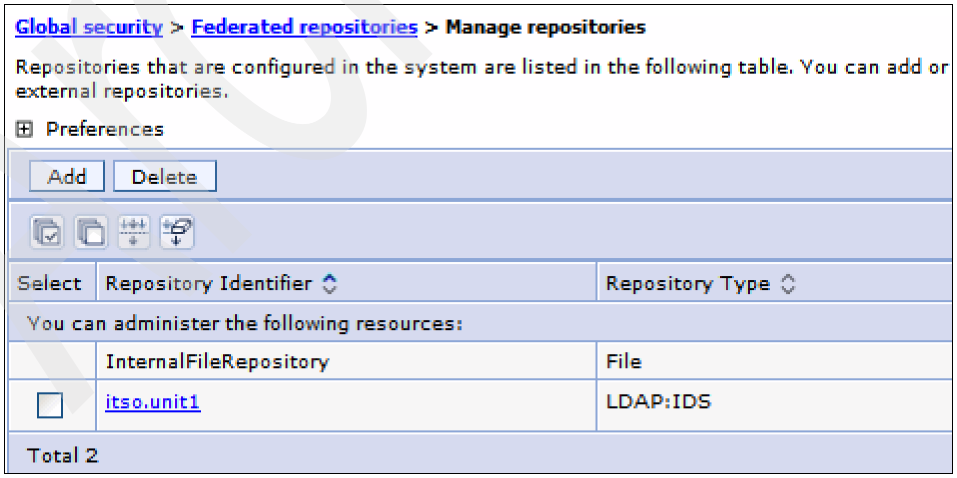
(IBM Tivoli Directory Server uses this attribute for all group mappings)

Click **OK**.

You have completed the configuration of the repository. These steps can be repeated in a variety of combinations to add additional LDAP repositories.

23. Click **OK** on the itso.unit1 panel.

24. Save the configuration.


Global security > Federated repositories > **Manage repositories**

Repositories that are configured in the system are listed in the following table. You can add or external repositories.

⊞ Preferences

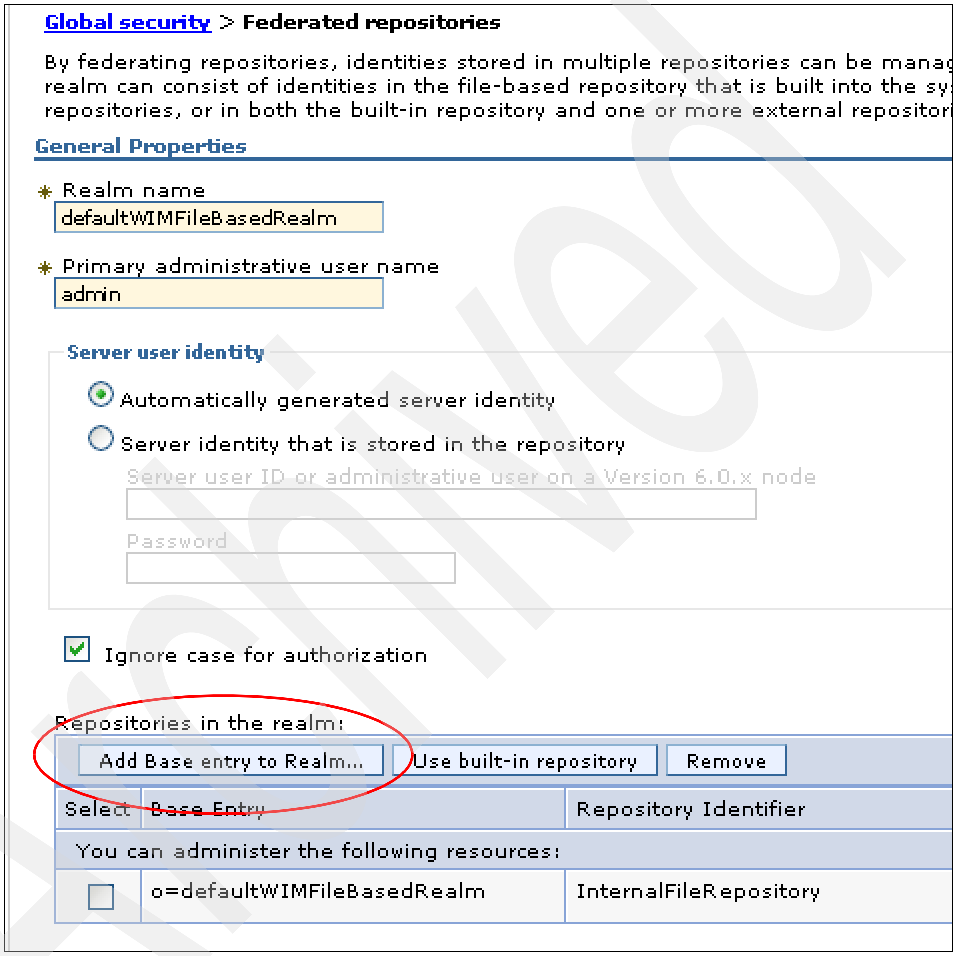
Add | Delete

Select	Repository Identifier	Repository Type
You can administer the following resources:		
	InternalFileRepository	File
☐	itso.unit1	LDAP:IDS
Total 2		

Figure 4-33 Global security → Federated repositories → Manage respositories panel

25. Click the **Federated repositories** link in the navigation trail in Figure 4-33 on page 115.

26. The next phase is to include the recently added managed repositories into the realm mapping by clicking **Add base entry to Realm** on the Federated repositories panel (refer to Figure 4-34).



Global security > Federated repositories

By federating repositories, identities stored in multiple repositories can be manag
realm can consist of identities in the file-based repository that is built into the sy
repositories, or in both the built-in repository and one or more external repositori

General Properties

* Realm name
defaultWIMFileBasedRealm

* Primary administrative user name
admin

Server user identity

[✓ Automatically generated server identity]

Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

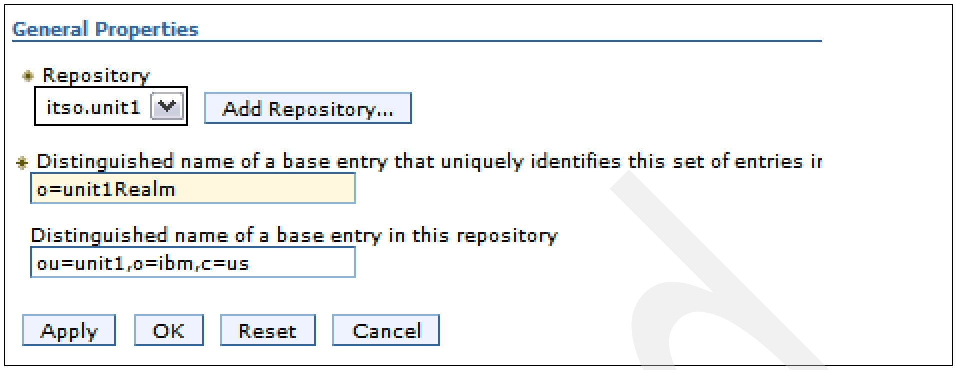
[✓ Ignore case for authorization]

Repositories in the realm:

Select	Base Entry	Repository Identifier
You can administer the following resources:		
☐	o=defaultWIMFileBasedRealm	InternalFileRepository

Figure 4-34 Add base entry to realm

27. The panel in Figure 4-35 on page 117 appears.


General Properties

Repository
itso.unit1

Add Repository...

Distinguished name of a base entry that uniquely identifies this set of entries ir
o=unit1Realm

Distinguished name of a base entry in this repository
ou=unit1,o=ibm,c=us

Apply OK Reset Cancel

Figure 4-35 Global security → Federated repositories → Repository reference

Complete the fields in this panel (Figure 4-35):

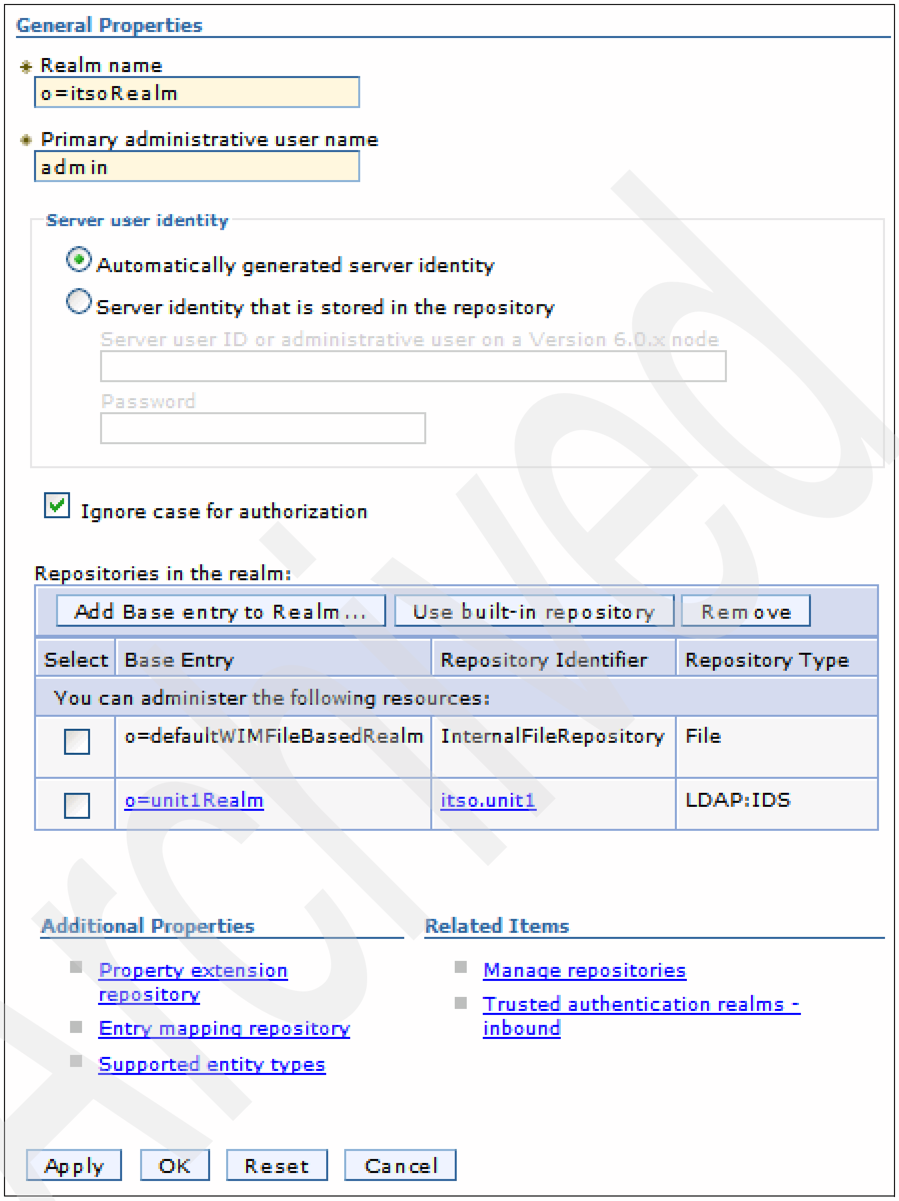
– Select the repository from the drop-down list: **itso.unit1**

– Enter the distinguished name of the base entry that uniquely identifies this set of entries in the realm.

– Enter the distinguished name of a base entry in this repository.

This distinguished name is the base entry of the LDAP directory.

28. Click **OK**.
29. (Optional) Click **Save.**
30. The next step is to complete the general properties for the federated repositories as shown in Figure 4-36 on page 118.


General Properties

Realm name
o=itsoRealm

Primary administrative user name
admin

Server user identity

[✓ Automatically generated server identity]

Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

[✓] Ignore case for authorization

Repositories in the realm:

Select	Base Entry	Repository Identifier	Repository Type
You can administer the following resources:			
☐	o=defaultWIMFileBasedRealm	InternalFileRepository	File
☐	o=unit1Realm	itso.unit1	LDAP:IDS

Figure 4-36 Global security → Federated respositories

In Figure 4-36:

– Specify a meaningful realm name. Unlike the stand-alone configurations, the federated repository allows for the explicit naming of the realm.

– Then, set the Primary administrative user ID. This user must exist in the default repository. The default repository here is a file repository if the repository was configured at installation. This field is the user that was specified when the file-based repository was created and typically is already populated. Otherwise, when completing this panel (Figure 4-36 on page 118), you will be prompted for the password.

> **Note:** The VMM default repository can be modified by changing the base entry for the default parent on each of the entity types, which you locate by navigating to the Supported entity types panel from the Additional Properties section of the Federated repositories panel.

Click **Apply**. If a prompt appears, enter a password and click **OK**.

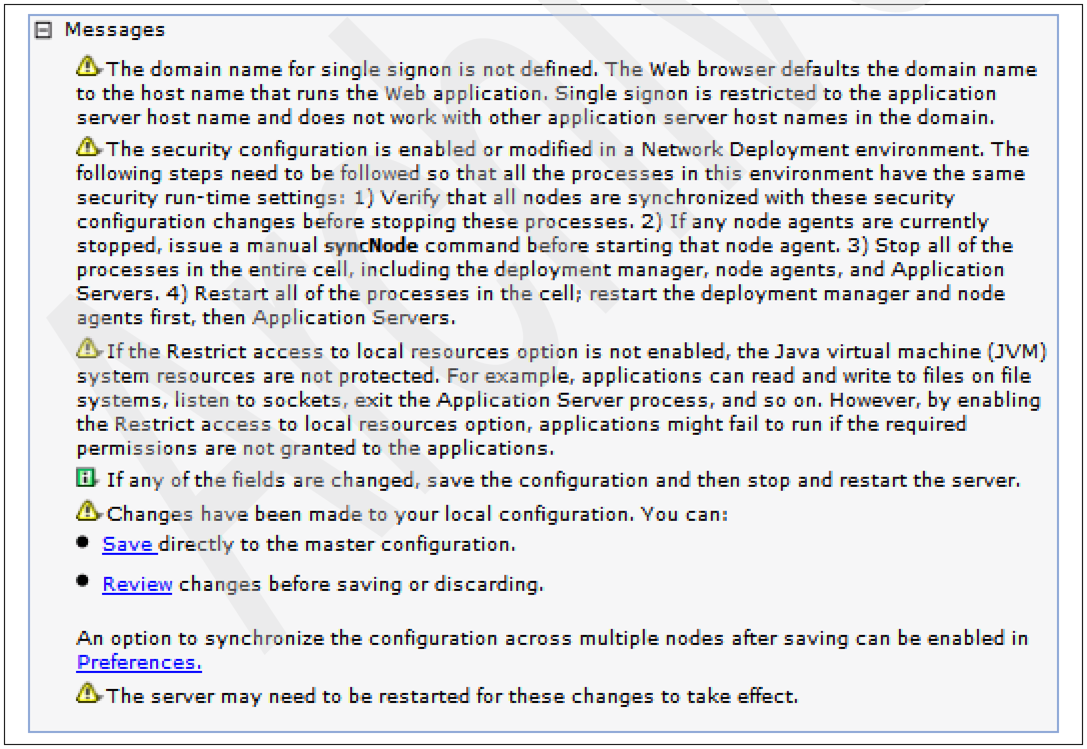
31. (Optional) Configure VMM database adapters for EMR or PER. For more information, refer to “Configuring VMM database base adapter features” on page 121.
32. Click **OK** on the Federated repositories panel, which will take you back to the Global security panel.
33. Save the configuration.
34. In the User account repository section of the Global security panel, verify that **Federated repositories** is selected in the Available realm definitions and click **Set as current** to the right of the field.
35. At this time, it is normal to configure any additional authentication and authorization preferences, which are further discussed in “Authentication and authorization settings” on page 133. For the continuation of this example, assume the steps have been completed.
36. After finalizing your security preferences and choosing which security types to activate, click **Apply** on the Global security panel.

> **Important:** Be sure to read the instructions supplied in the messages information (Figure 4-37 on page 120) regarding restarting and synchronizing nodes in a Network Deployment environment. These instructions are here to remind you that you must synchronize and restart your cell in order to complete the configuration.
>
> After restarting the environment, you can authenticate using the primary administration ID and then proceed to configure administrative security (refer to Chapter 2, “Administrative security” on page 17).

Tip: If you are unable to authenticate, you can turn off security to help troubleshoot the problem by running the following **wsadmin** commands (command in jacl format). *You must run this command locally; it cannot be executed remotely.*

Executed from <*profile*>/bin directory where <*profile*> is the profile's install directory.

```
wsadmin -conntype NONE
WASX7357I: By request, this scripting client is not connected to any
server process. Certain configuration and application operations
will be available in local mode.
WASX7029I: For help, enter: "$Help help"
wsadmin>securityoff
LOCAL OS security is off now but you need to restart server1 to make
it affected.
wsadmin>$AdminConfig save
wsadmin>exit
```



⊟ Messages

⚠ The domain name for single signon is not defined. The Web browser defaults the domain name to the host name that runs the Web application. Single signon is restricted to the application server host name and does not work with other application server host names in the domain.

⚠ The security configuration is enabled or modified in a Network Deployment environment. The following steps need to be followed so that all the processes in this environment have the same security run-time settings: 1) Verify that all nodes are synchronized with these security configuration changes before stopping these processes. 2) If any node agents are currently stopped, issue a manual **syncNode** command before starting that node agent. 3) Stop all of the processes in the entire cell, including the deployment manager, node agents, and Application Servers. 4) Restart all of the processes in the cell; restart the deployment manager and node agents first, then Application Servers.

⚠ If the Restrict access to local resources option is not enabled, the Java virtual machine (JVM) system resources are not protected. For example, applications can read and write to files on file systems, listen to sockets, exit the Application Server process, and so on. However, by enabling the Restrict access to local resources option, applications might fail to run if the required permissions are not granted to the applications.

ℹ If any of the fields are changed, save the configuration and then stop and restart the server.

⚠ Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

An option to synchronize the configuration across multiple nodes after saving can be enabled in Preferences.

⚠ The server may need to be restarted for these changes to take effect.

Figure 4-37 Global security: Click Save and take note of additional messages and warnings

37. Save the configuration.

4.3.5 Configuring VMM database base adapter features

In addition to supporting the default adapters for LDAP, file-based, and database repositories, the VMM also has adapters for the Property Extension Repository (PER) and Entry Mapping Repository (EMR). Refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.wim.doc.en/adapters.html

Configuring an EMR

The entry mapping repository is a database repository. It is used to store data that VMM needs to manage profiles on multiple repositories that are entry-level-joined.

An *entry-level join* means that the federated repository configuration uses multiple repositories simultaneously and recognizes the entries in the various repositories as entries representing distinct entities. For example, a company might have a Lightweight Directory Access Protocol (LDAP) directory that contains entries for its employees and a database that contains entries for business partners and clients. By configuring an entry mapping repository, a federated repository configuration can use both the LDAP and the database at the same time.

To configure an EMR:

1. Create the EMR database tables as shown in Example 4-10.

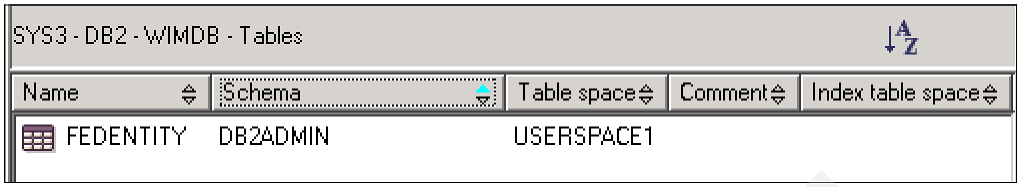
   The jacl format of the command is:

   ```
   $AdminTask setupIdMgrEntryMappingRepositoryTables{...}
   ```

Example 4-10 Create EMR tables

```
wsadmin>$AdminTask setupIdMgrEntryMappingRepositoryTables
{-schemaLocation "C:\WebSphere\AppServer\etc\wim\setup" -databaseType
db2 -dbURL jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB -dbAdminId
db2admin -dbDriver com.ibm.db2.jcc.DB2Driver-dbAdminPassword db2pwd01
-reportSqlError true}
CWWIM5099I Command completed successfully.
```

2. The execution of the command creates the database table that is shown in Figure 4-38 on page 122.


SYS3 - DB2 - WIMDB - Tables

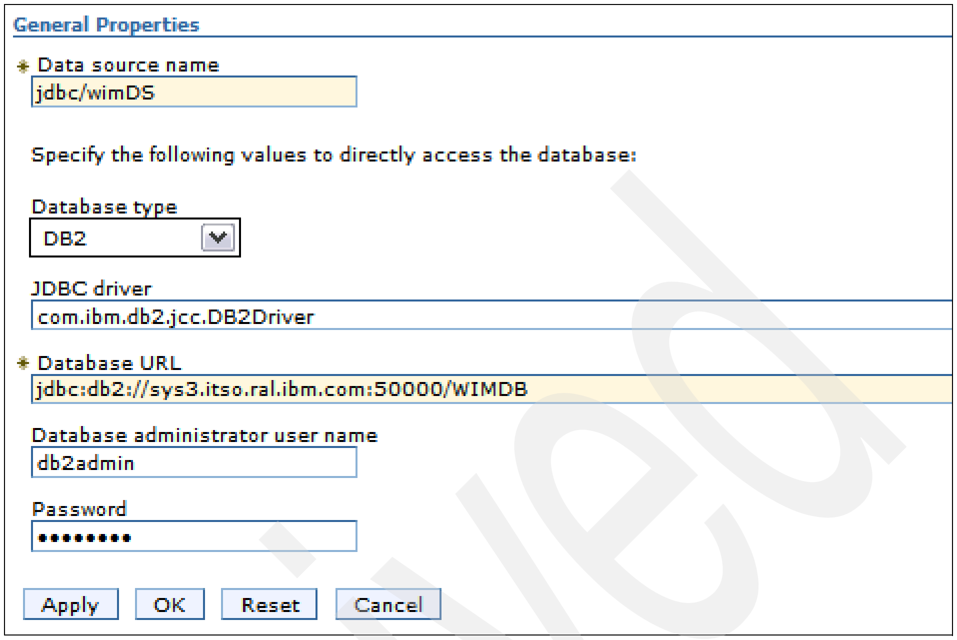
Name	Schema	Table space	Comment	Index table space
FEDENTITY	DB2ADMIN	USERSPACE1		

Figure 4-38 Entry mapping repository table

3. (Optional but recommended) Create the Java Database Connectivity (JDBC™) data source for the chosen database with the default Java Naming and Directory Interface (JNDI) name `jdbc/wimDS`. Because the federated repository is cell-scoped, the data source scope is cell also. If the data source is not found, the dbURL specified on the EMR window is defaulted.

Note: We do not provide further details about the creation of a data source here, because creating a data source is a typical administration task that is not related specifically to security.

4. Navigate to the Global security panel. Select Federated repositories in the drop-down for the available realm definitions and click **Configure.** Select **Entry mapping repository** in the Additional Properties section of the panel.
5. Complete the database information as shown in Figure 4-39 on page 123.


General Properties

* Data source name: jdbc/wimDS

Specify the following values to directly access the database:

Database type: DB2

JDBC driver: com.ibm.db2.jcc.DB2Driver

* Database URL: jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB

Database administrator user name: db2admin

Password: ••••••••

Figure 4-39 Global security → Federated respositories → Entry mapping repository panel

Enter these values on this panel:

- Data source name: `jdbc/wimDS`
- Database type: `DB2`
- JDBC driver: `com.ibm.db2.jcc.DB2Driver`
- Database URL: `jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB`
- Database administrator user name: `db2admin`
- Password: `db2pwd01`

Click **OK** and save the configuration.

Configuring a property extension repository

For security and business reasons, you might not want to allow write operations to your repositories. However, applications calling the federated repository configuration might need to store additional properties for the entities. A federated repository configuration provides a *property extension repository (PER)*, which is a database, regardless of the type of main profile repositories, for a property-level join configuration. For example, a company that uses an LDAP directory for its internal employees and a database for external customers and business partners might not allow write access to its LDAP and its database. The company can use the property extension repository in a federated repository configuration to store additional properties for the people in those repositories, excluding the user ID. When an application uses the federated repository configuration to retrieve an entry for a person, the federated repository configuration transparently joins the properties of the person that are retrieved from either the LDAP or the client's database with the properties of the person that are retrieved from the property extension repository into a single logical person entry.

To create the PER database tables:

1. Execute a **wsadmin** command as shown in Example 4-11.

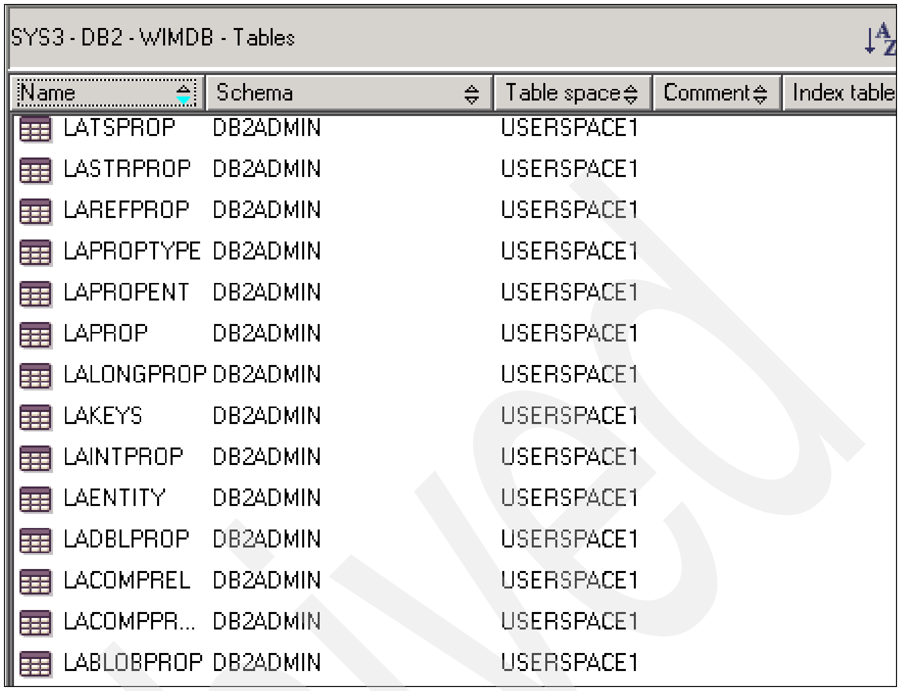
   The jacl format is:

   ```
   $AdminTask setupIdMgrPropertyExtensionRepositoryTables{...}
   ```

Example 4-11 Create property extension repository tables

```
wsadmin>$AdminTask setupIdMgrPropertyExtensionRepositoryTables
{-schemaLocation "C:\WebSphere\AppServer\etc\wim\setup" -laPropXML
"C:\WebSphere\AppServer\etc\wim\setup\wimlaproperties.xml"
-databaseType db2 -dbURL jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB
-dbAdminId db2admin -dbDriver com.ibm.db2.jcc.DB2Driver
-dbAdminPassword db2pwd01 -reportSqlError true}
CWWIM5099I Command completed successfully.
```

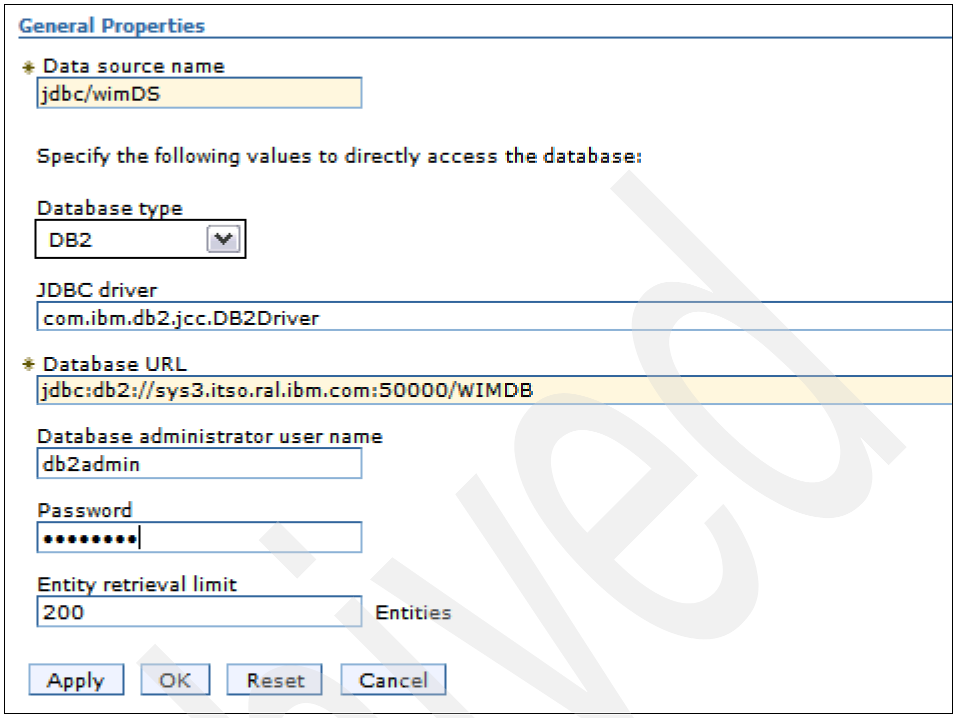
2. The execution of the **wsadmin** command creates the database tables that are shown in Figure 4-40 on page 125.


SYS3 - DB2 - WIMDB - Tables

Name	Schema	Table space	Comment	Index table
LATSPROP	DB2ADMIN	USERSPACE1		
LASTRPROP	DB2ADMIN	USERSPACE1		
LAREFPROP	DB2ADMIN	USERSPACE1		
LAPROPTYPE	DB2ADMIN	USERSPACE1		
LAPROPENT	DB2ADMIN	USERSPACE1		
LAPROP	DB2ADMIN	USERSPACE1		
LALONGPROP	DB2ADMIN	USERSPACE1		
LAKEYS	DB2ADMIN	USERSPACE1		
LAINTPROP	DB2ADMIN	USERSPACE1		
LAENTITY	DB2ADMIN	USERSPACE1		
LADBLPROP	DB2ADMIN	USERSPACE1		
LACOMPREL	DB2ADMIN	USERSPACE1		
LACOMPPR...	DB2ADMIN	USERSPACE1		
LABLOBPROP	DB2ADMIN	USERSPACE1		

Figure 4-40 Property extension repository tables

3. (Optional but recommended) Create the JDBC data source for the chosen database with the default JNDI name `jdbc/wimDS`. Because the federated repository is cell-scoped, the data source scope is cell-scoped also. If the data source is not found, the dbURL specified on the EMR window is defaulted.
4. Navigate to the Federated repositories panel and select **Property extension repository** in the Additional Properties section of the panel.
5. Complete the database information (Figure 4-41 on page 126).


General Properties

Data source name: jdbc/wimDS

Specify the following values to directly access the database:

Database type: DB2

JDBC driver: com.ibm.db2.jcc.DB2Driver

Database URL: jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB

Database administrator user name: db2admin

Password: ••••••••

Entity retrieval limit: 200 Entities

Figure 4-41 Properties extension repository panel

Enter the attributes:

- Data source name: `jdbc/wimDS`
- Database type: `DB2`
- JDBC driver: `com.ibm.db2.jcc.DB2Driver`
- Database URL: `jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB`
- Database administrator user name: `db2admin`
- Password: `db2pwd01`
- Entity retrieval limit: `200 (default)`

6. Click **OK** and save the configuration.

4.3.6 Configuring elements of federated repositories using wsadmin

You can also configure the same elements of the LDAP federated repository using the **`wsadmin`** and the scripting environment.

Example 4-1 on page 82 shows the commands that can be executed to configure the same federated repository. These commands use the jython scripting language and were executed using **`wsadmin`** in an interactive mode. (We captured the commands by using the administrative console Command Assistance).

Note: For simplicity, authentication settings, such as the authentication mechanism and SSO enablement, are not handled here. Thus, these script examples are a minimal set of commands using the authentication and authorization defaults. For example, the setAdminActiveSecuritySettings command can have many more options.

Configuring an LDAP repository

To configure the LDAP repository:

1. Create a managed LDAP repository configuration that defines the LDAP that WebSphere Application Server will federate (Example 4-12).

Example 4-12 Create LDAP repository

```
# Create the repository LDAP repository
AdminTask.createIdMgrLDAPRepository('[-default true -id itso.unit1
-ldapServerType IDS -sslConfiguration -certificateMapMode exactdn
-certificateFilter -loginProperties uid]')

# Modify the general properties
AdminTask.addIdMgrLDAPServer('[-id itso.unit1 -host
sys4.itso.ral.ibm.com -bindDN
uid=admin1,ou=adminusers,ou=unit1,o=ibm,c=us -bindPassword
admin1pwd01-authentication simple -referal ignore -sslEnabled false
-ldapServerType IDS -sslConfiguration -certificateMapMode exactdn
-certificateFilter -port 389]')

AdminTask.deleteIdMgrLDAPAttr('[-id itso.unit1 -name -entityTypes
PersonAccount]')

AdminTask.addIdMgrLDAPAttr('[-id itso.unit1 -propertyName kerberosId
-name -entityTypes PersonAccount]')

# Add failover server
```

```
AdminTask.listIdMgrLDAPBackupServers('[-id itso.unit1 -primary_host
sys4.itso.ral.ibm.com]')

AdminTask.addIdMgrLDAPBackupServer('[-id itso.unit1 -primary_host
sys4.itso.ral.ibm.com -host sys1.itso.ral.ibm.com -port 389]')

AdminConfig.save()
```

2. Modify the LDAP Entity Type mapping. In Example 4-13, the script modifies the supported object classes for the Group entity type.

Example 4-13 Modify LDAP Group entity type

```
AdminTask.updateIdMgrLDAPEntityType('[-id itso.unit1 -name Group
-objectClasses groupOfNames;groupOfUniqueNames;groupOfURLs -searchBases
-searchFilter ]')

AdminConfig.save()
```

3. Most LDAP directories have a standard attribute on every person object that identifies the groups of which the object is a member. This attributes varies from directory to directory. Certain directories do not support this attribute; in which case, the application server needs to search all of the group types to identify the user group relationship. For IBM Tivoli Directory Server, the attribute is `ibm-allGroups`. Example 4-14 shows how to modify the group attribute definition. This script identifies the group membership attribute and maps the group attributes that are identified by this attribute.

Example 4-14 Identify the LDAP group membership attribute

```
AdminTask.addIdMgrLDAPGroupMemberAttr('[-id itso.unit1 -name
uniqueMember -objectClass groupOfUniqueNames -scope direct]')

AdminTask.addIdMgrLDAPGroupDynamicMemberAttr('[-id itso.unit1 -name
memberURL -objectClass groupOfURLs]')

AdminTask.setIdMgrLDAPGroupConfig('[-id itso.unit1 -name ibm-allGroups
-scope all]')

AdminConfig.save()
```

4. Configure the base entry for the LDAP repository, and add the base entry to the realm (Example 4-15 on page 129).

Example 4-15 Add base entry and add entry to realm

```
AdminTask.addIdMgrRepositoryBaseEntry('[-id itso.unit1 -name 
o=unit1Realm -nameInRepository ou=unit1,o=ibm,c=us]')

AdminTask.addIdMgrRealmBaseEntry('[-name o=itsoRealm -baseEntry 
o=unit1Realm]')

AdminConfig.save()
```

5. Validate the registry, set the registry type, and enable administrative security (Example 4-16).

Example 4-16 Validate and enable security

```
AdminTask.validateAdminName('[-registryType WIMUserRegistry -adminUser 
admin ]')

AdminTask.setAdminActiveSecuritySettings('[-activeUserRegistry 
WIMUserRegistry -enableGlobalSecurity true]')

AdminConfig.save()
```

Configuring VMM database-based support features

Important: The script commands demonstrated here exclude the script steps for creating the databases that are documented in 4.3.5, “Configuring VMM database base adapter features” on page 121. We also do not show the creation of referenced data sources.

To configure the VMM database-based support features:

1. Configure EMR (Example 4-17).

Example 4-17 Configure EMR

```
AdminTask.setIdMgrEntryMappingRepository('[-dataSourceName jdbc/wimDS 
-databaseType db2 -dbURL jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB 
-dbAdminId db2admin -dbAdminPassword db2pwd01-JDBCDriverClass 
com.ibm.db2.jcc.DB2Driver]')
```

2. Configure PER (Example 4-18).

Example 4-18 Configure PER

```
AdminTask.setIdMgrPropertyExtensionRepository('[-dataSourceName 
jdbc/wimDS -databaseType db2 -dbURL
jdbc:db2://sys3.itso.ral.ibm.com:50000/WIMDB -dbAdminId db2admin
-dbAdminPassword db2pwd01-JDBCDriverClass com.ibm.db2.jcc.DB2Driver
-entityRetrievalLimit 200]')
```

4.3.7 Configuring a database repository in VMM

Important: The following example shows you how to configure a database repository. We do not recommend the use of a database for the selected repository type. The recommended industry best practice is to use an LDAP directory.

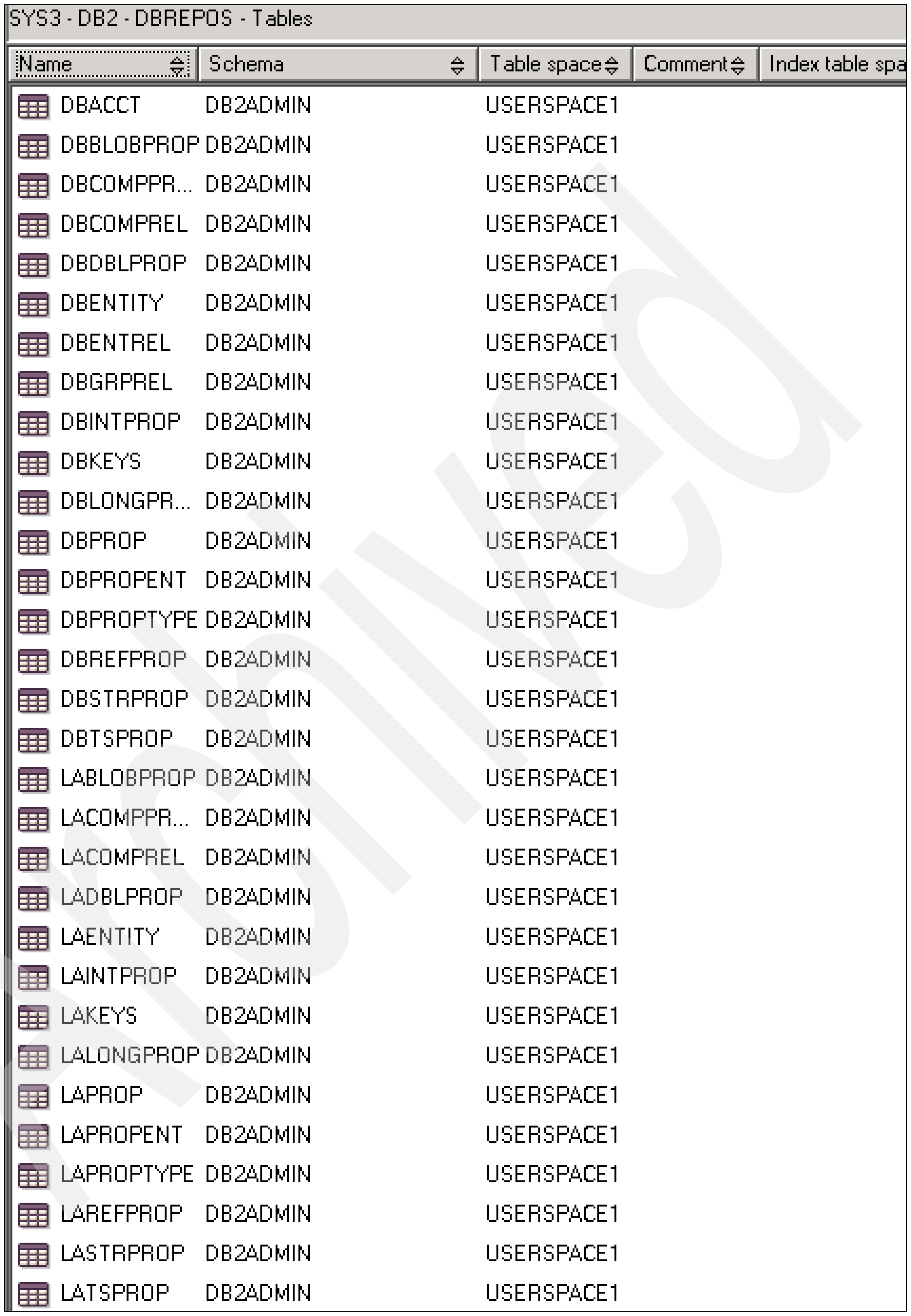
You need to execute a series of wsadmin tasks to configure the repository. This section provides an example of the configuration steps:

1. Create the database repository tables by executing a **`wsadmin`** command as shown in Example 4-19 (jacl format `$AdminTask setupIdMgrDBTables {...}`).

Example 4-19 Create database repository tables: $AdminTask setupIdMgrDBTables

```
wsadmin>$AdminTask setupIdMgrDBTables {-schemaLocation
"C:\WebSphere\AppServer\etc\wim\setup" -dbPropXML
"C:\WebSphere\AppServer\etc\wim\setup\wimdbproperties.xml"
-databaseType db2 -dbURL jdbc:db2://sys3.itso.ral.ibm.com:50000/DBREPOS
-dbAdminId db2admin -dbDriver com.ibm.db2.jcc.DB2Driver
-dbAdminPassword db2pwd01 -reportSqlError true}
CWWIM5099I Command completed successfully.
```

2. If you use a database other than the database used by PER and EMR, you create the tables that are shown in Figure 4-42 on page 131 (the tables for PER and EMR repositories can be in the same database).



SYS3 - DB2 - DBREPOS - Tables

Name	Schema	Table space	Comment	Index table spa
DBACCT	DB2ADMIN	USERSPACE1		
DBBLOBPROP	DB2ADMIN	USERSPACE1		
DBCOMPPR...	DB2ADMIN	USERSPACE1		
DBCOMPREL	DB2ADMIN	USERSPACE1		
DBDBLPROP	DB2ADMIN	USERSPACE1		
DBENTITY	DB2ADMIN	USERSPACE1		
DBENTREL	DB2ADMIN	USERSPACE1		
DBGRPREL	DB2ADMIN	USERSPACE1		
DBINTPROP	DB2ADMIN	USERSPACE1		
DBKEYS	DB2ADMIN	USERSPACE1		
DBLONGPR...	DB2ADMIN	USERSPACE1		
DBPROP	DB2ADMIN	USERSPACE1		
DBPROPENT	DB2ADMIN	USERSPACE1		
DBPROPTYPE	DB2ADMIN	USERSPACE1		
DBREFPROP	DB2ADMIN	USERSPACE1		
DBSTRPROP	DB2ADMIN	USERSPACE1		
DBTSPROP	DB2ADMIN	USERSPACE1		
LABLOBPROP	DB2ADMIN	USERSPACE1		
LACOMPPR...	DB2ADMIN	USERSPACE1		
LACOMPREL	DB2ADMIN	USERSPACE1		
LADBLPROP	DB2ADMIN	USERSPACE1		
LAENTITY	DB2ADMIN	USERSPACE1		
LAINTPROP	DB2ADMIN	USERSPACE1		
LAKEYS	DB2ADMIN	USERSPACE1		
LALONGPROP	DB2ADMIN	USERSPACE1		
LAPROP	DB2ADMIN	USERSPACE1		
LAPROPENT	DB2ADMIN	USERSPACE1		
LAPROPTYPE	DB2ADMIN	USERSPACE1		
LAREFPROP	DB2ADMIN	USERSPACE1		
LASTRPROP	DB2ADMIN	USERSPACE1		
LATSPROP	DB2ADMIN	USERSPACE1		

Figure 4-42 Database repository tables

3. Create the managed repository as shown in Example 4-20.

   The jacl format is: `$AdminTask createIdMgrDBRepository`

Example 4-20 Create DB repository: $AdminTask createIdMgrDBRepositroy

```
wsadmin>$AdminTask createIdMgrDBRepository {-id DB2Repos
-dataSourceName jdbc/wimDBRepos -databaseType db2 -dbURL
jdbc:db2://sys3.itso.ral.ibm.com:50000/DBREPOS -JDBCDriverClass
com.ibm.db2.jcc.DB2Driver -dbAdminId db2admin -dbAdminPassword
db2pwd01}
CWWIM5046W Each configured repository must contain at least one base
entry. Add a base entry before saving the configuration. For LDAP
repository, add the LDAP server before adding the base entry.
```

4. Configure the base entry for the repository as shown in Example 4-21.

   The jacl format is: `$AdminTask addIdMgrRepositoryBaseEntry`

Example 4-21 Configure base entry: $AdminTask addIdMgrRepositoryBaseEntry

```
wsadmin>$AdminTask addIdMgrRepositoryBaseEntry {-id DB2Repos -name
"o=database.org"}
CWWIM5028I  The configuration is saved in a temporary workspace. You
must use the "$AdminConfig save" command to save it in the master
repository.
```

5. Add the base entry into the realm (`o=itsoRealm` that was created earlier) as shown in Example 4-22.

   The jacl format is: `$AdminTask addIdMgrRealmBaseEntry`

Example 4-22 Add base entry to realm: $AdminTask addIdMgrRealmBaseEntry

```
wsadmin>$AdminTask addIdMgrRealmBaseEntry { -name "o=itsoRealm"
-baseEntry "o=database.org"}
CWWIM5028I  The configuration is saved in a temporary workspace. You
must use the "$AdminConfig save" command to save it in the master
repository.
```

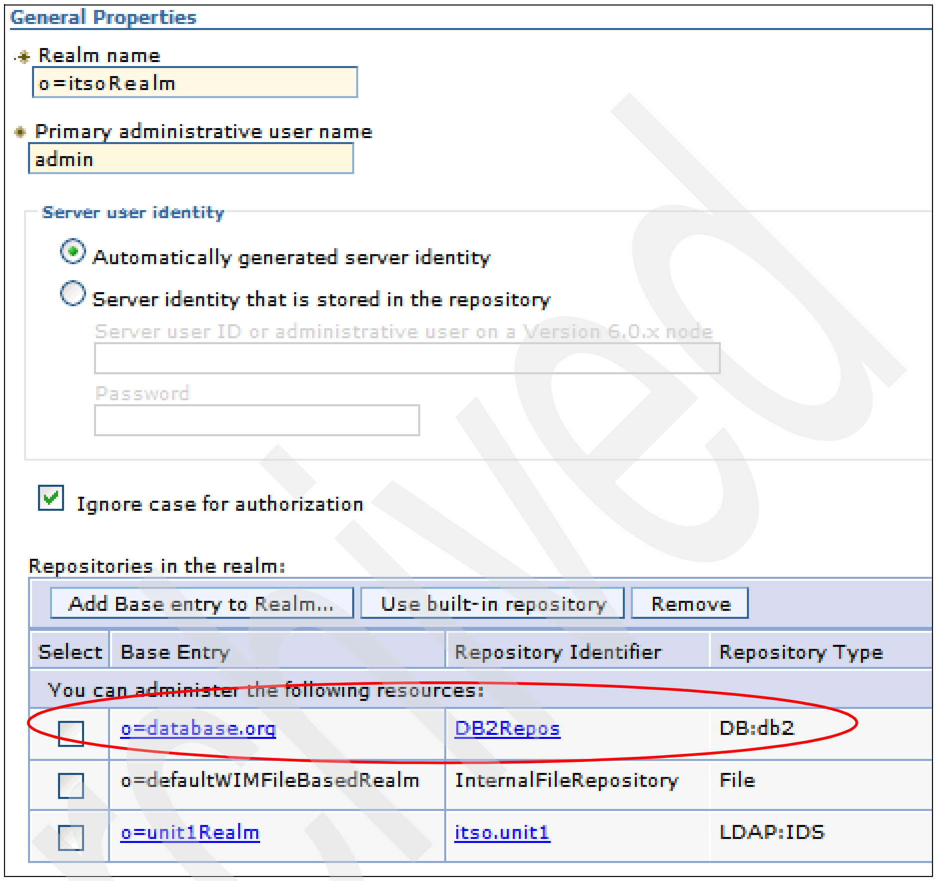
6. Save the configuration as shown in Example 4-23.

Example 4-23 Save Configuration: $AdminConfig save

```
wsadmin>$AdminConfig save

wsadmin>
```

7. The result of the configuration (Figure 4-43) can be seen in the administrative console in the Federated repositories page.


General Properties

Realm name: o=itsoRealm

Primary administrative user name: admin

Server user identity

(●) Automatically generated server identity

( ) Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

[✓] Ignore case for authorization

Repositories in the realm:

Select	Base Entry	Repository Identifier	Repository Type
You can administer the following resources:			
☐	o=database.org	DB2Repos	DB:db2
☐	o=defaultWIMFileBasedRealm	InternalFileRepository	File
☐	o=unit1Realm	itso.unit1	LDAP:IDS

Figure 4-43 Global security → Federated repositories: Database repository added

4.4 Authentication and authorization settings

Authentication and authorization in the application server occur when a request is made on a secured resource. In a Web container, this request might be a servlet, an image, or another Web resource. In an EJB container, this request is an EJB method.

If a user makes an initial request on a communication channel (HTTP, Remote Method Invocation (RMI)/Internet Inter-ORB Protocol (IIOP), SOAP/HTTP, and so forth) and if that request is for a secured resource, the user needs to provide authentication information. If the request is on a channel that supports prompting, such as HTTP running Web applications, the user will be prompted for the the required authentication information. The required authentication information varies depending on the channel and transport configurations in the application server. The key is that the identity of the user making the request needs to be validated.

Additionally, most systems today deploy a form of single sign-on (SSO), so that after a user is authenticated, the user can access all of the systems that the user is permitted to access without needing to provide user authentication information again. There are typically three types of SSO implementations:

- A common security infrastructure that is supported by a single token mechanism, which is the model of LTPA and Kerberos. LTPA is the WebSphere Application Server SSO token, and it is the default SSO mechanism. Kerberos is an industry token for SSO.
- Identity assertion models where systems are configured to trust the authentication that was validated on other systems. This mechanism asserts the user identity without actually supplying the user authentication token (that is, the password). The mechanism relies on a trust between systems. This approach has many forms. Two examples of standards that are supported by WebSphere Application Server are Common Security Interoperability Version 2 (CSIv2) and certain token types that are used in WS-Security.
- You can implement SSO using a re-authentication model. This approach requires passing the user and its authentication token to the downstream system.

After a user is authenticated, whether via SSO or a password supplied in a Web page authentication in a Web application, the application server will authorize the user to verify that the user of the validated identity is permitted access to the requested resource.

A simple example of the core components involved in authentication for a trust token authentication flow is shown in Figure 4-44 on page 135. The purpose of this diagram is to help you to begin to think about and consider the authentication and authorization choices that need to be made before and when enabling security.

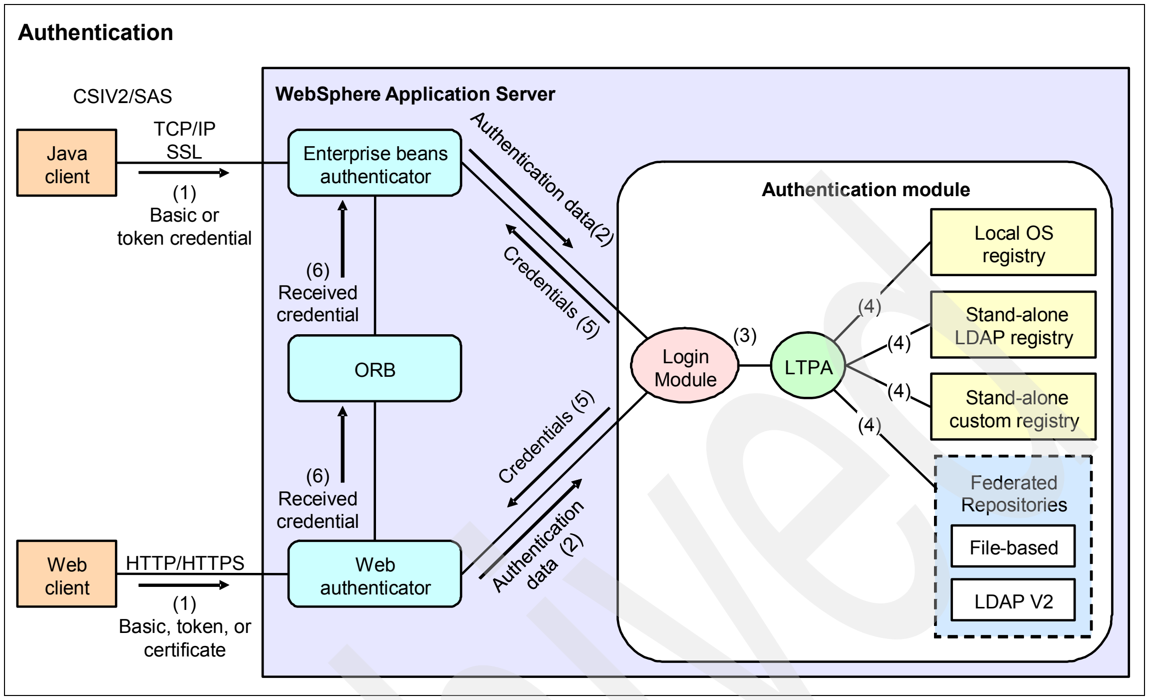


Figure 4-44 WebSphere Application Server simplified authentication component diagram

Figure 4-44 illustrates the following simplified authentication flow (these numbers correlate to the numbers in Figure 4-44:

1. A request arrives on an input channel (for example, Web or EJB).
2. The authentication data is passed through the authentication modules.
3. This flow highlights the WebSphere Application Server default trust association interceptor (TAI) for LTPA token. LTPA is the application server's default and the recommended trust token implementation. Trust is asserted at this stage of the processing. If the token is determined valid, the user information that it contains is trusted and the identity of the user is asserted.
4. After token processing, the user credentials are rebuilt from the information that is retrieved from the token.
5. User credentials are created based on user information retrieved from the registry.
6. The credential is forwarded as the request is processed by different architectural tiers of the application server.

Now, in interpreting the diagram in Figure 4-44 on page 135, it is useful to identify the security decisions that have been made in relation to authentication and authorization. This list summarizes the implied and explicit security decisions that were made with regard to Figure 4-44 on page 135:

- Web requests are participating in a form of SSO, implied by the passing of the token on the request.

  Security decision: Enable SSO.

- The token of choice for SSO is LTPA.

  Security decision: Choose an authentication mechanism.

- The use of a token authentication mechanism indicates establishing a trust domain.

  Security decision: Determine the boundaries of the trust.

- EJB client requests are secured, and EJB clients use CSIv2 communications.

  Security decision: Secure RMI/IIOP communications.

- SSL communications are used on both RMI/IIOP and Web-based communications.

  Security decision: Secure transports.

- Identity attributes are propagated between channels.

  Security decision: Propagate identity.

- Having authenticated the request, authorization checking will have been processed. Because there is no indication of an external authorization provider, it can be implied that the default internal authorization provider was used.

  Security decision: Use default authorization provider.

The decisions that were made for the environment that is represented by Figure 4-44 on page 135 are the typical decisions that need to be made when establishing each security domain in the cell. Table 4-3 on page 137 summarizes these decision points in the form of configuration questions that need to be answered to begin to configure the authentication and authorization settings of a security domain.

Table 4-3 Summary of security questions for authentication and authorization

Decision item	Security questions	References
1	Do transport-based communications need to be secured?	"SSL communications" on page 137 Chapter 5, "Secure Sockets Layer administration" on page 151
2	What servers and domains are trusted by this security domain?	"Extending the trust domain" on page 138 Chapter 3, "Using security domains" on page 51
3	Will this server need to forward the identity to another server in a trust relationship?	"Web and SIP security" on page 139 "RMI/IIOP security" on page 141
4	What are the authentication mechanisms that are used by this security domain?	"Authentication section defaults of the Global security panel" on page 138 "JAAS login modules" on page 145
5	What is the authorization mechanism that is used by this security domain?	"External authorization" on page 145

4.4.1 Identifying key authentication and authorization defaults

WebSphere Application Server has default authentication and authorization settings. The following sections provide a summary of the defaults and highlights areas where these defaults can be expanded or customized. The security domain that is used in this example is the global or cell domain.

SSL communications

SSL communications are a special case, and they have specialized configuration requirements. We discuss SSL communications in more detail in Chapter 5, "Secure Sockets Layer administration" on page 151.

But generally, this question needs to be answered each time that a configuration for outbound or inbound communications is made. If a communication is made to a system outside of the application server process, the communication transport must be secured.

Typical examples and communications that have SSL communications include:

- HTTP communications, for example, HTTP Client to Web servers and Web server plug-in to application server, inbound Web service communications, and remote portal communications
- Application server to registry implementation, for example, an application server communicating with an LDAP registry
- Application server to application server, for example, RMI/IIOP (EJB) communications and Web services.

Less common uses of SSL communications but equally possible are:

- Application server to secured data resources over JDBC
- Enterprise information system (EIS) or secured messaging communications

Your organization’s security policy determines which communications need to be secured.

Extending the trust domain

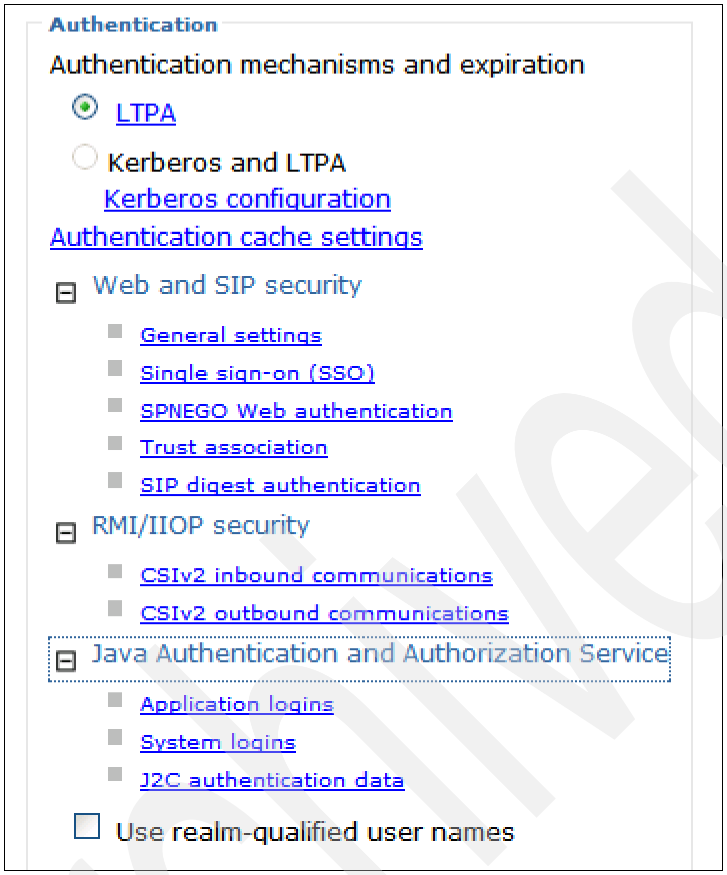
In a security domain, it is sometimes desirable to expand the trust of a domain by including other realms. Be cautious when you extend the trust domain. Security administrators are typically adverse to increasing the trust of a domain.

The addition of a realm to the trust scope means that LTPA tokens from a trusted realm will be permitted to assert a user’s identity. Requests from the external realm, if subsequently authorized, will be permitted to execute work in the trusting security domain. The trusted realms at the cell level are specified as a function of the user registry configuration. For other security domains, it is configured as part of the domain’s configuration.

By default, other domains are not trusted.

Authentication section defaults of the Global security panel

The authentication section of the Global security panel shows the items to consider when setting the authentication mechanisms (Figure 4-45 on page 139).


Authentication

Authentication mechanisms and expiration

● LTPA

○ Kerberos and LTPA

Kerberos configuration

Authentication cache settings

⊟ Web and SIP security

- General settings
- Single sign-on (SSO)
- SPNEGO Web authentication
- Trust association
- SIP digest authentication

⊟ RMI/IIOP security

- CSIv2 inbound communications
- CSIv2 outbound communications

⊟ Java Authentication and Authorization Service

- Application logins
- System logins
- J2C authentication data

☐ Use realm-qualified user names

Figure 4-45 Authentication section of Global security panel

Note in Figure 4-45 that LTPA is the default token type and authentication mechanism when using token-based identity assertion.

Note: Kerberos must be configured before the Kerberos option can be selected. You use the Kerberos configuration link.

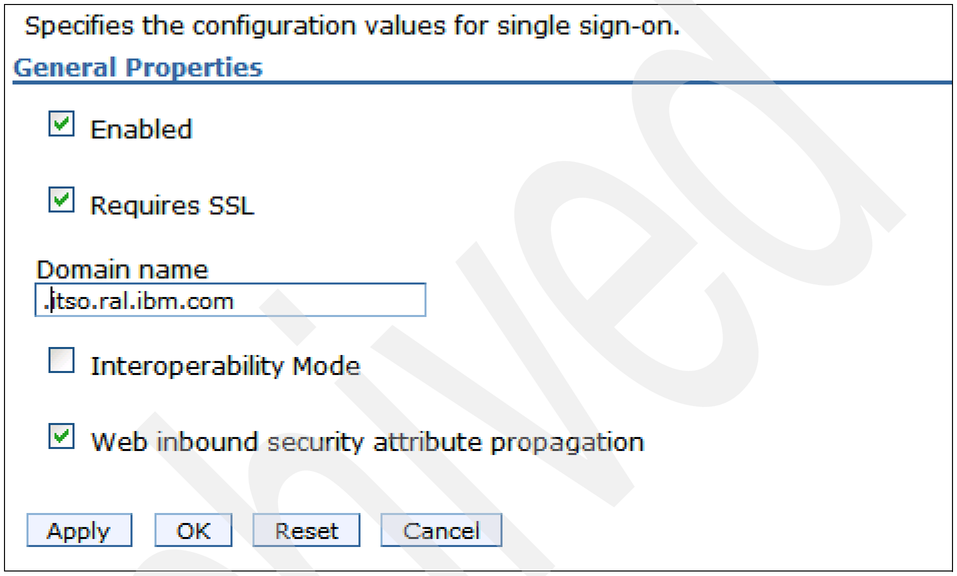
Web and SIP security

The Web and Session Initiation Protocol (SIP) security configuration settings consist of several links to configuration pages that allow you to adjust security settings for Web application access.

If you select **Single sign-on (SSO)** to enable SSO, we recommend you use SSL communication to protect the token confidentiality, but SSL communication is not the default. To enforce SSL for SSO, select **Requires SSL** in the single sign-on

(SSO) configuration settings (Figure 4-46). This option assumes that other SSL configuration requirements have been satisfied. Note that the propagation of token attributes is the default behavior.

The other configuration entry that must be configured for SSO is the Domain name field. This field contains a delimited list that defines the host URLs that are included in the SSO domain.


Specifies the configuration values for single sign-on.

General Properties

✓ Enabled

✓ Requires SSL

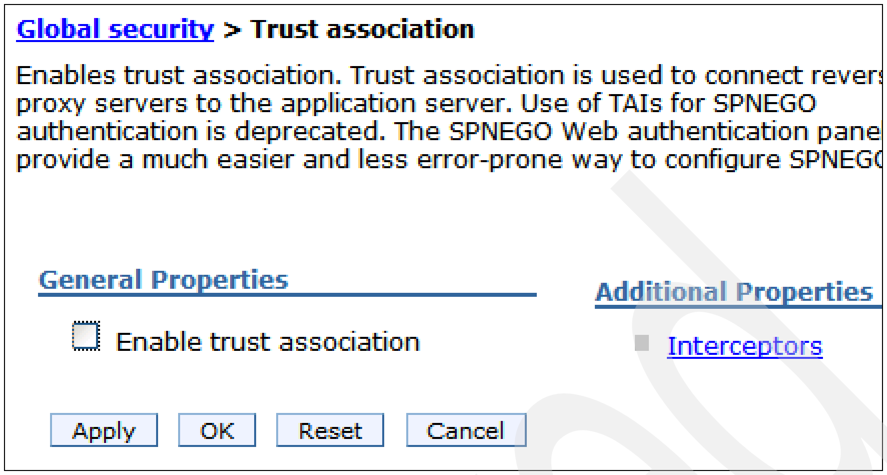
Domain name: .itso.ral.ibm.com

☐ Interoperability Mode

✓ Web inbound security attribute propagation

Figure 4-46 Single sign-on (SSO) settings

If authentication proxies provide the authentication before requests are forwarded to the Web container, it is possible to activate the security domain trust association located in the Trust association settings (Figure 4-47 on page 141). Trust interceptors are the preferred authentication mechanism for Web authentication.


Global security > Trust association

Enables trust association. Trust association is used to connect revers
proxy servers to the application server. Use of TAIs for SPNEGO
authentication is deprecated. The SPNEGO Web authentication pane
provide a much easier and less error-prone way to configure SPNEG

General Properties

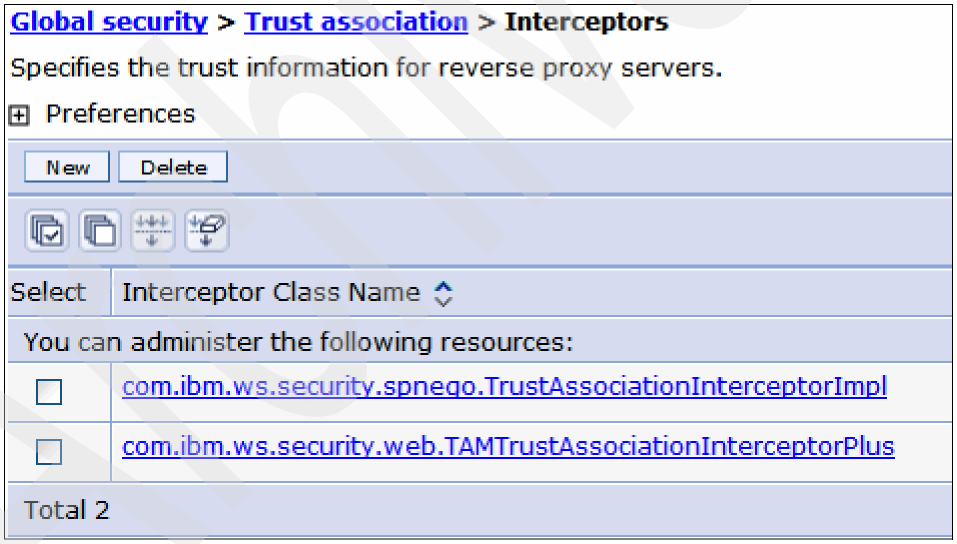
☐ Enable trust association

Additional Properties

- Interceptors

Figure 4-47 Trust association settings

The interceptors configured by default are shown in Figure 4-48.


Global security > Trust association > Interceptors

Specifies the trust information for reverse proxy servers.

Select	Interceptor Class Name
You can administer the following resources:	
☐	com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl
☐	com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus
Total 2	

Figure 4-48 Trust association default interceptors

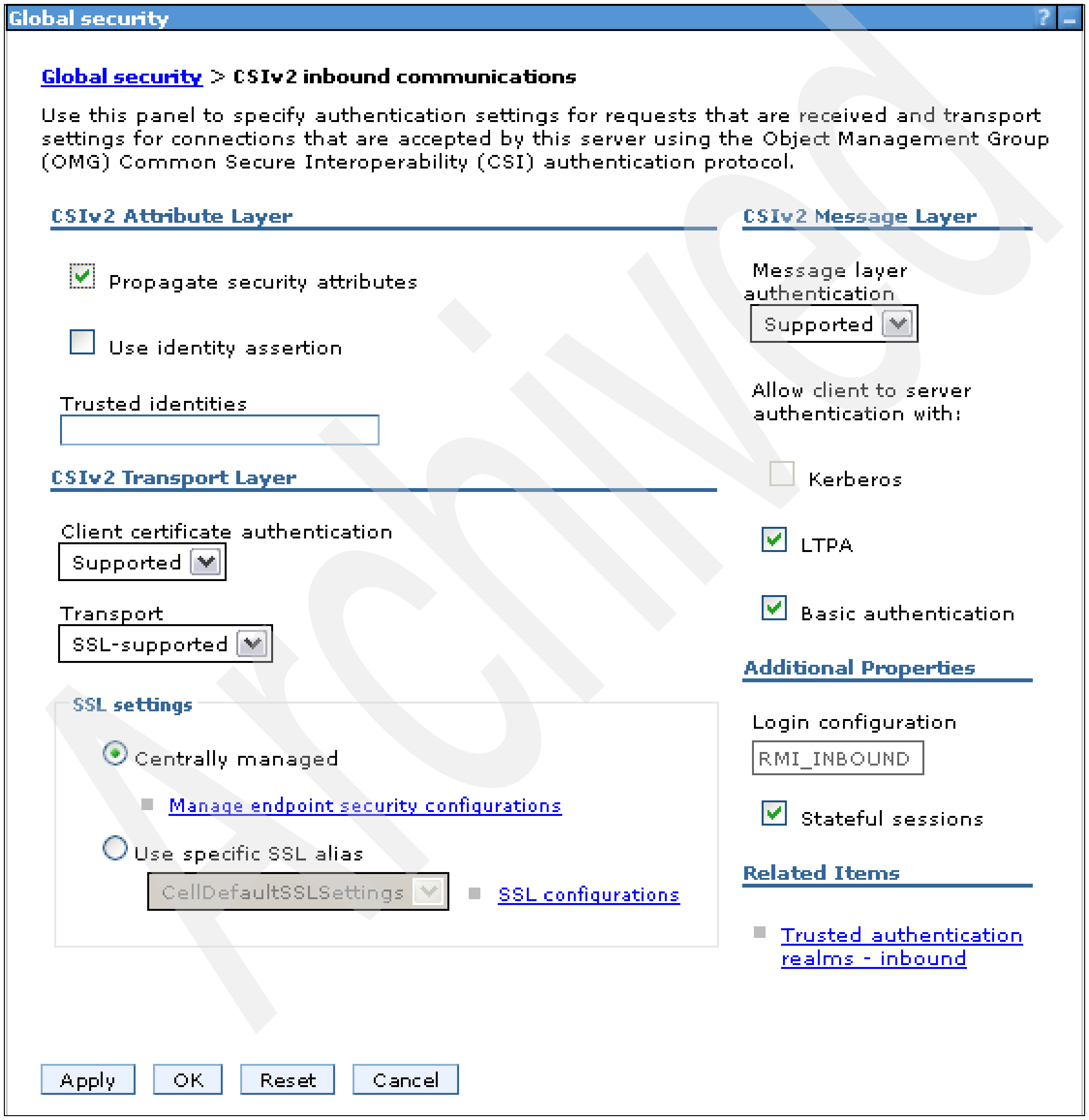
RMI/IIOP security

The defaults for the RMI/IIOP security include:

- Support SSL, which means that the WebSphere Application Server by default will use SSL for RMI/IIOP if the connecting client initiates the connection using SSL. By default, SSL is not enforced.

- Propagation of identity attributes is enabled
- Message layer support for Basic or LTPA authentication mechanisms

To see the defaults for the inbound configuration, select **CSIv2 inbound communications** from the Global security page. The defaults are shown in Figure 4-49.



Global security > CSIv2 inbound communications

Use this panel to specify authentication settings for requests that are received and transport settings for connections that are accepted by this server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

CSIv2 Attribute Layer

[✓] Propagate security attributes

[ ] Use identity assertion

Trusted identities:

CSIv2 Transport Layer

Client certificate authentication: Supported

Transport: SSL-supported

SSL settings

(●) Centrally managed

- Manage endpoint security configurations

( ) Use specific SSL alias: CellDefaultSSLSettings
- SSL configurations

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

[ ] Kerberos

[✓] LTPA

[✓] Basic authentication

Additional Properties

Login configuration: RMI_INBOUND

[✓] Stateful sessions

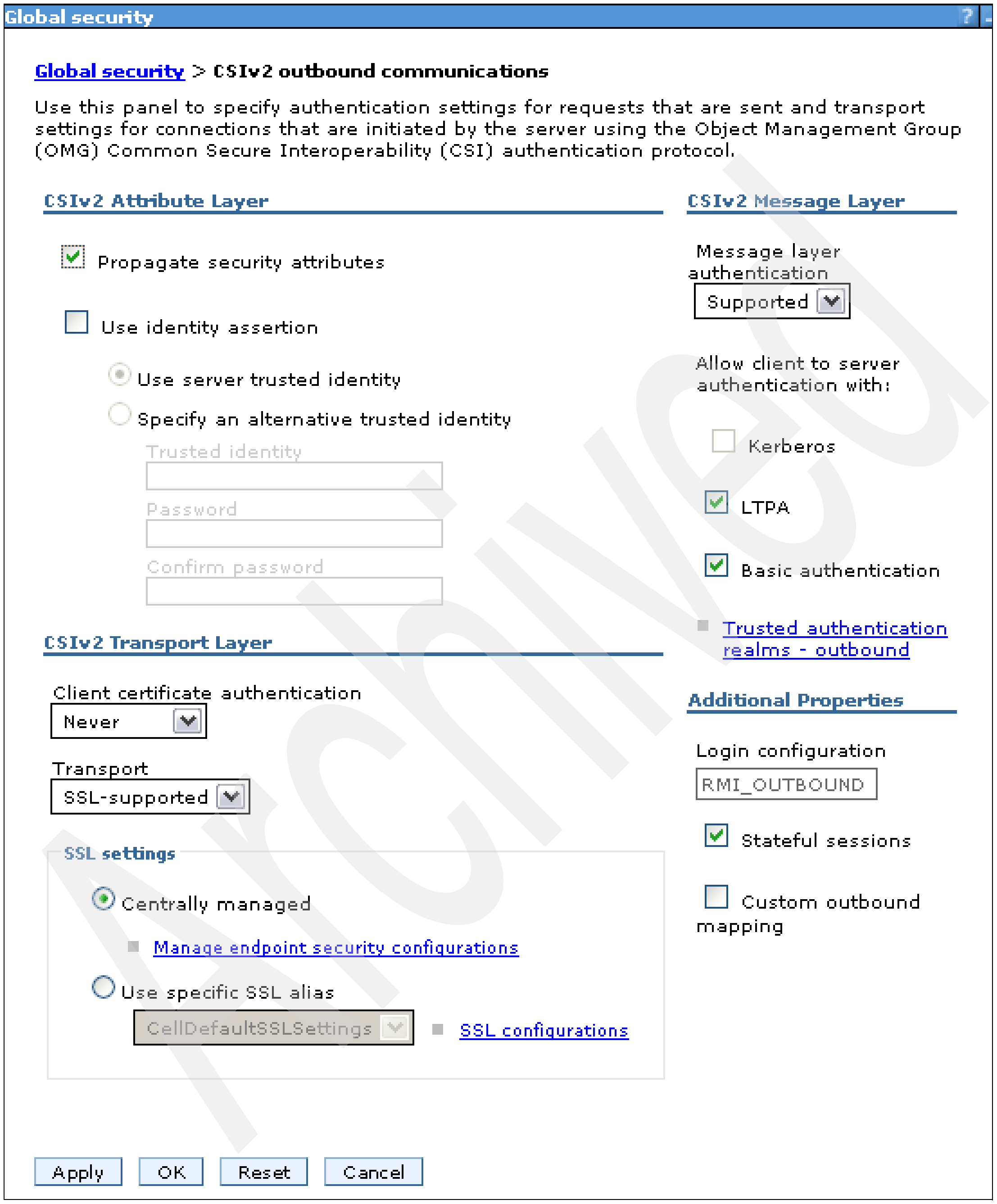
Related Items

- Trusted authentication realms - inbound

Figure 4-49 CSIv2 inbound communications

To see the outbound configuration defaults, select **CSIv2 outbound communications** from the Global security page.

The outbound configuration defaults are shown in Figure 4-50 on page 144.


Global security

Global security > CSIv2 outbound communications

Use this panel to specify authentication settings for requests that are sent and transport settings for connections that are initiated by the server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

CSIv2 Attribute Layer

- [✓] Propagate security attributes
- [ ] Use identity assertion
  - (●) Use server trusted identity
  - ( ) Specify an alternative trusted identity
    - Trusted identity:
    - Password:
    - Confirm password:

CSIv2 Transport Layer

Client certificate authentication: Never

Transport: SSL-supported

SSL settings

- (●) Centrally managed
  - Manage endpoint security configurations
- ( ) Use specific SSL alias: CellDefaultSSLSettings — SSL configurations

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

- [ ] Kerberos
- [✓] LTPA
- [✓] Basic authentication
- Trusted authentication realms - outbound

Additional Properties

Login configuration: RMI_OUTBOUND

- [✓] Stateful sessions
- [ ] Custom outbound mapping

Apply | OK | Reset | Cancel

Figure 4-50 CSIv2 outbound communications

The most common and recommended change here is to have CSIv2 inbound and outbound enforce SSL by changing the Transport from **SSL-supported** to **SSL-required**.

JAAS login modules

The login configurations that are mapped for the various logins are an important part of the authentication and authorization framework. For example, the CSIv2 defaults utilize the RMI_INBOUND and RMI_OUTBOUND login configurations for inbound and outbound CSIv2 communications. The Information Center provides details of the default system and application login configurations:

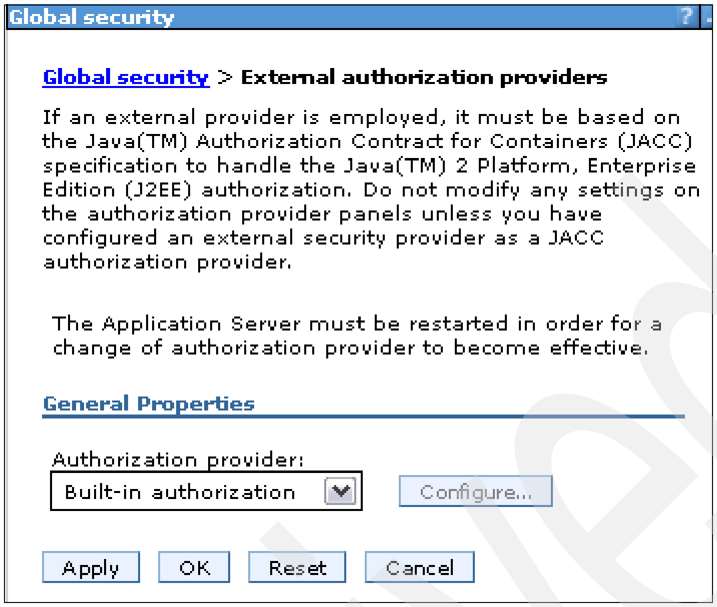
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/uwbs_jaasconfig.html

The defaults are normally more than sufficient for most needs. If you are considering customizing login modules, we recommend that you seek IBM or IBM Business Partner guidance to understand the security impacts of this choice.

External authorization

Selecting **External authorization providers** on the Global security page opens the panel that defines whether authorization decisions are deferred to an external Java Authorization Contract for Containers (JACC) specification authorization provider or are handled by the internal authorization provider.

The default is a built-in internal authorization provider. The built-in authorization provider gathers its user and group information from the configured user registry. Figure 4-51 on page 146 shows the built-in authorization as the authorization provider.



Global security > External authorization providers

If an external provider is employed, it must be based on the Java(TM) Authorization Contract for Containers (JACC) specification to handle the Java(TM) 2 Platform, Enterprise Edition (J2EE) authorization. Do not modify any settings on the authorization provider panels unless you have configured an external security provider as a JACC authorization provider.

The Application Server must be restarted in order for a change of authorization provider to become effective.

General Properties

Authorization provider: Built-in authorization

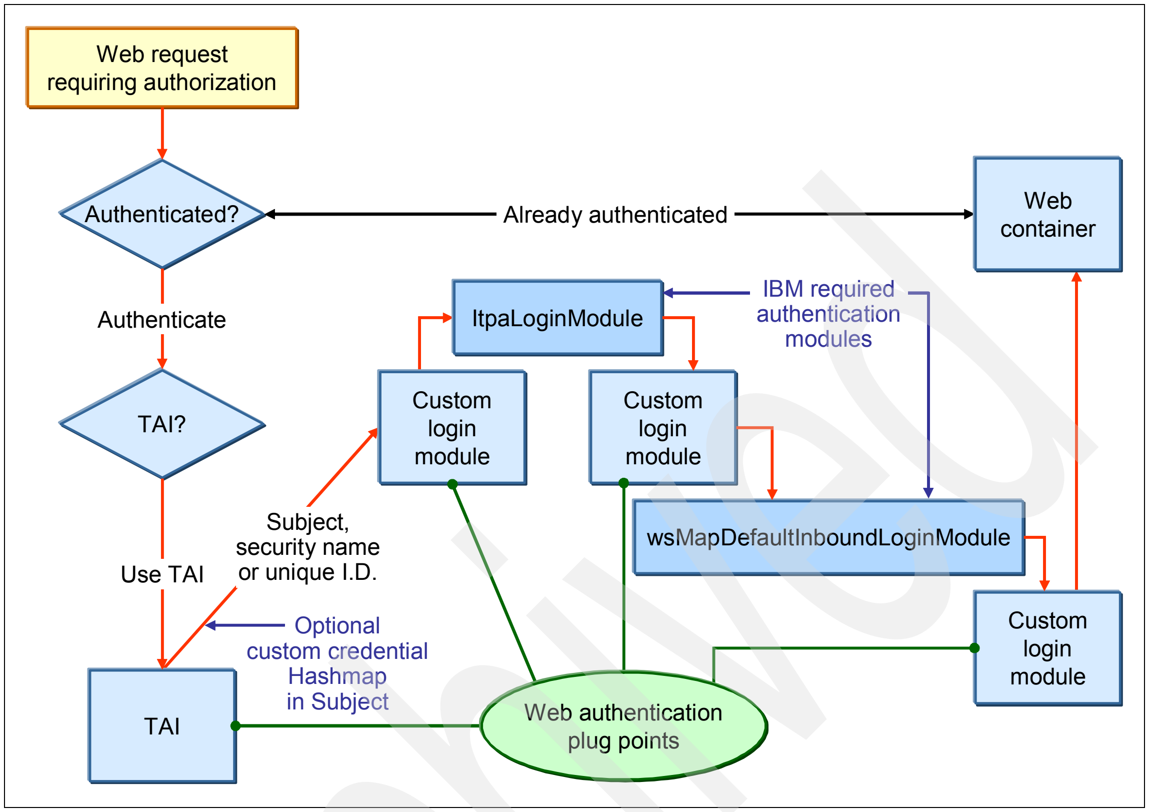
Figure 4-51 External authorization providers

4.4.2 Custom authentication choices

The JAAS security framework is inherently extensible and has many plug-in points for custom login modules. Similarly, custom interceptors can also be written.

For example, consider the three request types of Web Inbound, RMI inbound, and RMI outbound. Each request type has pluggable configuration points where custom authentication modules can be inserted.

Figure 4-52 on page 147 shows the pluggable points for Web inbound communications.



Web request requiring authorization
Authenticated?
Already authenticated
Web container
Authenticate
ItpaLoginModule
IBM required authentication modules
TAI?
Custom login module
Custom login module
wsMapDefaultInboundLoginModule
Subject, security name or unique I.D.
Use TAI
Optional custom credential Hashmap in Subject
Custom login module
TAI
Web authentication plug points

Figure 4-52 Web Inbound custom authentication points

Figure 4-53 on page 148 shows the pluggable points for RMI inbound communications.

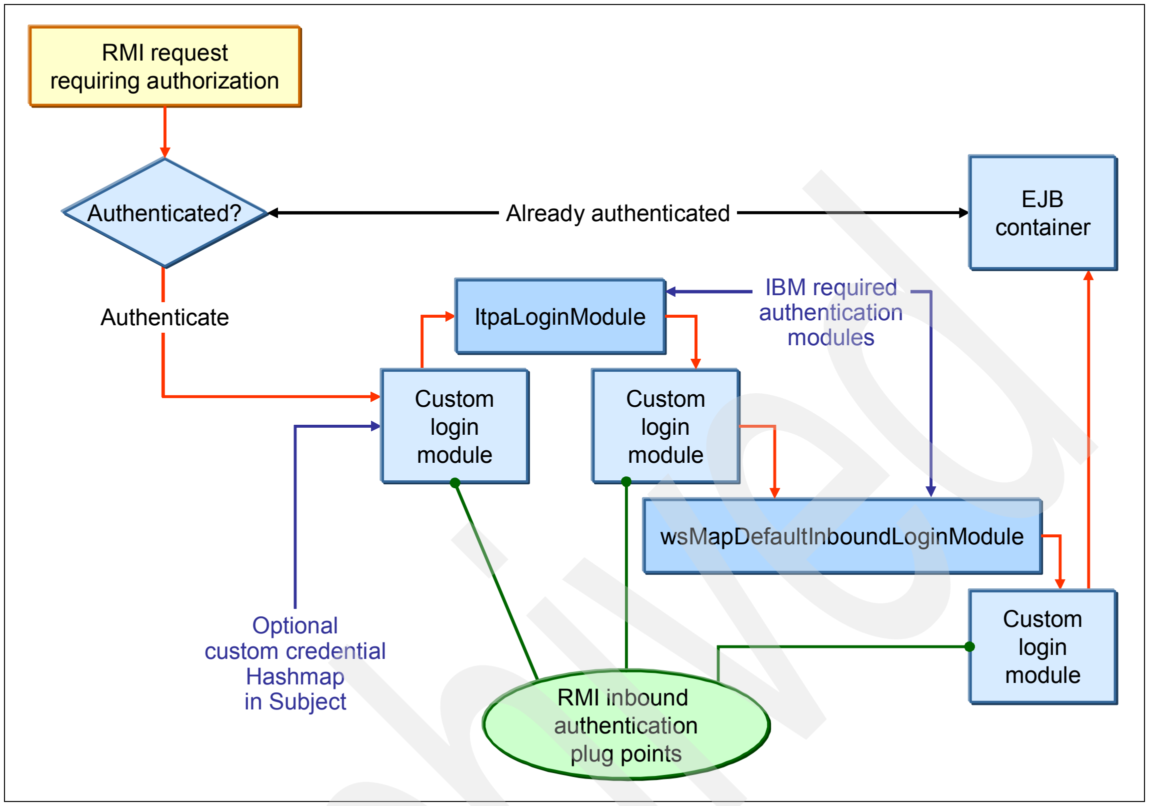

RMI request requiring authorization
Authenticated?
Already authenticated
EJB container
Authenticate
ItpaLoginModule
IBM required authentication modules
Custom login module
Custom login module
wsMapDefaultInboundLoginModule
Custom login module
Optional custom credential Hashmap in Subject
RMI inbound authentication plug points

Figure 4-53 RMI inbound custom authentication points

Figure 4-54 on page 149 shows the pluggable points for RMI outbound communications.

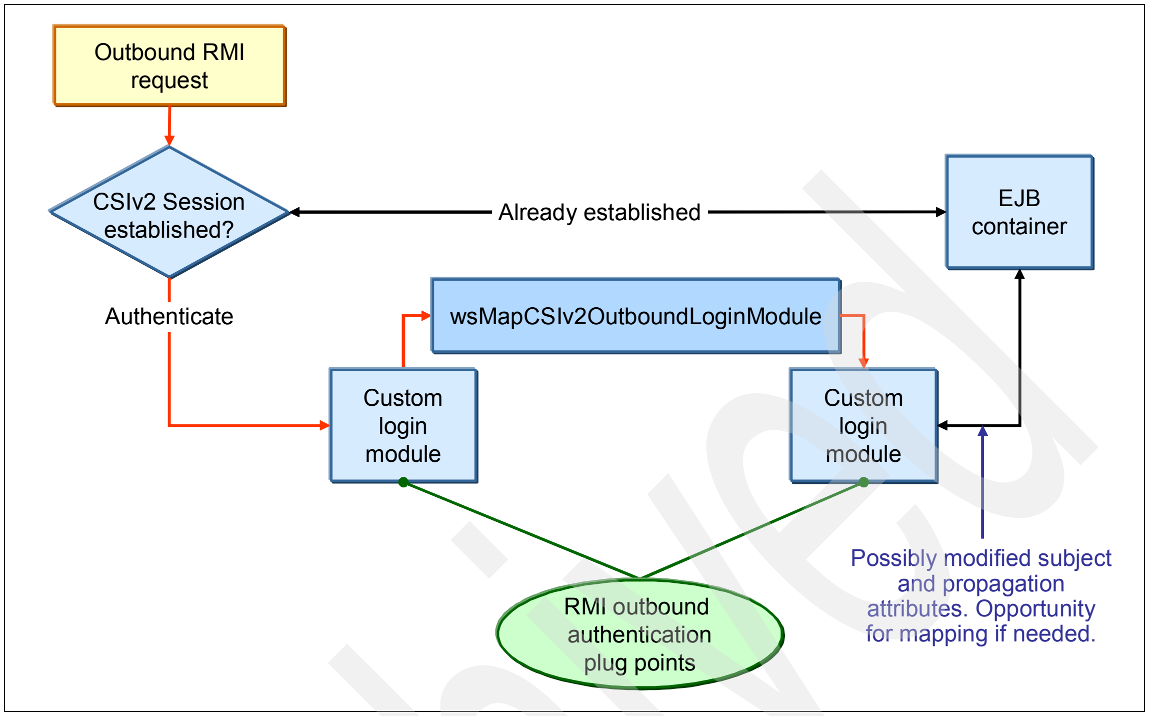

Outbound RMI request
CSIv2 Session established?
Already established
EJB container
Authenticate
wsMapCSIv2OutboundLoginModule
Custom login module
Custom login module
RMI outbound authentication plug points
Possibly modified subject and propagation attributes. Opportunity for mapping if needed.

Figure 4-54 RMI outbound custom authentication points

For more information about configuring custom TAIs, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsip_cfgsecuctai.html

For more information about configuring custom login modules, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_jaascustlogmod.html

Tip: Writing custom security in itself is not always difficult, but writing custom code that is secure can be difficult and is typically complicated. We recommend that any custom implementation include the following considerations:

- Confirm that the code being implemented scales and meets performance requirements.
- Consider the implications of how secure the implementation is, such as, for a trust interceptor, is the trust validation secure and can it truly be trusted. Or, is it possible to forge or impersonate the trust validation?
- Custom security must be independently evaluated regarding its reliability in relation to securing the environment.
- Remember to make infrastructure code, such as security, manageable by including features, such as logging.
- Generally, the key recommendation is to seek out experts and specialists in security for guidance.

5

Secure Sockets Layer administration

WebSphere Application Server utilizes the Secure Sockets Layer (SSL) protocol to provide transport layer security that ensures a secure connection between a client and server. WebSphere Application Server provides a full set of mechanisms for managing SSL configuration.

This chapter will show how to implement SSL in a WebSphere Application Server environment. It contains the following topics:

5.1 Secure communications using SSL

SSL is the industry standard for data interchange encryption between clients and servers. The foundation technology for SSL is public key cryptography. It uses a combination of asymmetric and symmetric key encryption models.

A series of messages, referred to as a *handshake*, are exchanged at the start of an SSL session. These messages allow the client to authenticate the server using public key techniques and, optionally, for the server to authenticate the client. The handshake uses asymmetric keys that consist of a public key and a private key. The public key can be distributed widely, but the private key is never distributed; it is always kept secret. When an entity encrypts data using a public key, only entities with the corresponding private key can decrypt that data.

During the handshake, the client and server cooperate in creating symmetric keys for speed and efficiency to use for further communication.

Java Secure Sockets Extension (JSSE) is used in WebSphere to handle the handshake negotiation and protection capabilities that are provided by SSL to ensure secure connectivity exists across most protocols. JSSE provides a Java specification for the implementation of the industry standard X.509 standard public key infrastructure (PKI).

A PKI represents a system of digital certificates, certificate authorities, registration authorities, a certificate management service, and a certification path validation algorithm. A PKI verifies the identity and the authority of each party that is involved in an Internet transaction, either financial or operational, with requirements for identity verification. It also supports the use of certificate revocation lists (CRLs), which are lists of revoked certificates.

An SSL configuration determines whether one entity can connect to the other entity and the peer connection that is trusted by an SSL handshake. If you do not have the necessary certificates, the handshake fails because the peer is not trusted. Figure 5-1 on page 153 shows a hypothetical keystore and truststore configuration example. Private keys are stored in the keystore file. The public keys are stored in the form of trusted certificates in the truststore.

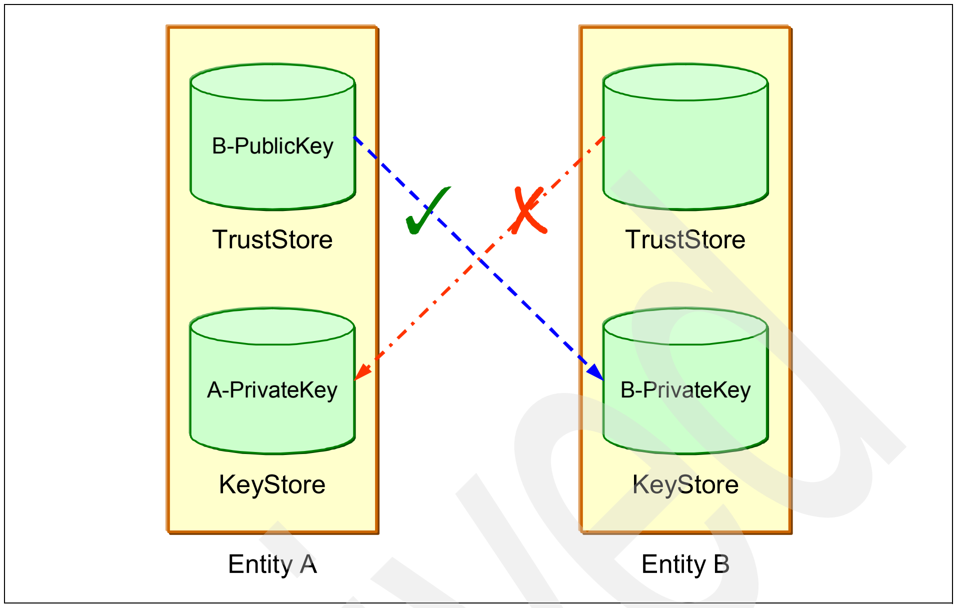


Figure 5-1 KeyStore and TrustStore

In Figure 5-1, the truststore for Entity A contains the public key of Entity B. Entity A can connect to Entity B, because Entity B can decrypt the data using its private key, but the truststore for Entity B does not contain the public key of Entity A. Therefore, if Entity A requires validation with Entity B, the handshake will fail.

You can think of Entity A as a client and Entity B as a server. If SSL is configured for server authentication, it will work in this scenario. But if SSL is configured for both client and server authentication, it will fail because there is no trust key on the server side to authenticate the client.

5.1.1 Certificates

Digitally signed X.509 certificates are used to establish SSL connections. Among the information you will find in an X.509 certificate is its distinguished name, validity dates, public key information, and the certificate signature.

The contents of a certificate are signed in one of these ways:

- Certificate authority (CA)

  A *certificate authority* is a trusted third-party organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate.

Server-side ports that accept connections from the general public must use CA-signed certificates. Most clients or browsers already have the signer certificate that can validate the X.509 certificate so signer exchange is not necessary for a successful connection.

- A root certificate in NodeDefaultRootStore or DmgrDefaultRootStore

  Certificates are issued in the form of a tree structure, with the root certificate at the top of the structure. Certificates signed by the root certificate inherit the trustworthiness of the certificate.

- Are self-signed

  Self-signed certificates are for use with a peer in a controlled environment, such as internal network communications. To complete a handshake, you must first send a copy of the entity certificate to every peer that connects to the entity.

When working with SSL connections in WebSphere, you will work with the following types of certificates:

- Personal certificates

  A personal certificate without a private key is a certificate that represents the entity that owns it during a handshake. Personal certificates contain both public and private keys.

- Signer certificates

  A signer certificate is a certificate that represents a peer entity or itself. Signer certificates contain just the public key and verify the signature of the identity that is received during a peer-to-peer handshake.

- Chained certificates *(new in V7)*

  A chained certificate is a personal certificate signed by a root certificate. Using a chained certificate enables you to refresh the personal certificate without affecting the trust established. It also enables tailoring of the certificate during profile creation (you can import your own or change the distinguished name (DN) of the one created by default) as well as the ability to change the default keystore password.

  Chained certificates replace the self-signed certificate. Signer certificates from the self-signed certificate that are distributed across the security configuration are replaced with the signer certificates from the root certificate that is used to sign the chained certificate.

  A default chained certificate is created at profile creation and stored in the default keystore. The root signer (public key) of the chained certificate is added to the default truststore.

5.1.2 Keystores and truststores

JSSE configurations typically reference a keystore and a truststore. By convention the keystore reference represents a Java keystore object that holds personal certificates and the truststore reference represents a Java keystore object that holds signer certificates. You may have only one store that holds both - personal certificates and signer certificates.

A default keystore and truststore are created by WebSphere Application Server during profile creation. You can also create a new keystore and truststores using the WebSphere administration tools.

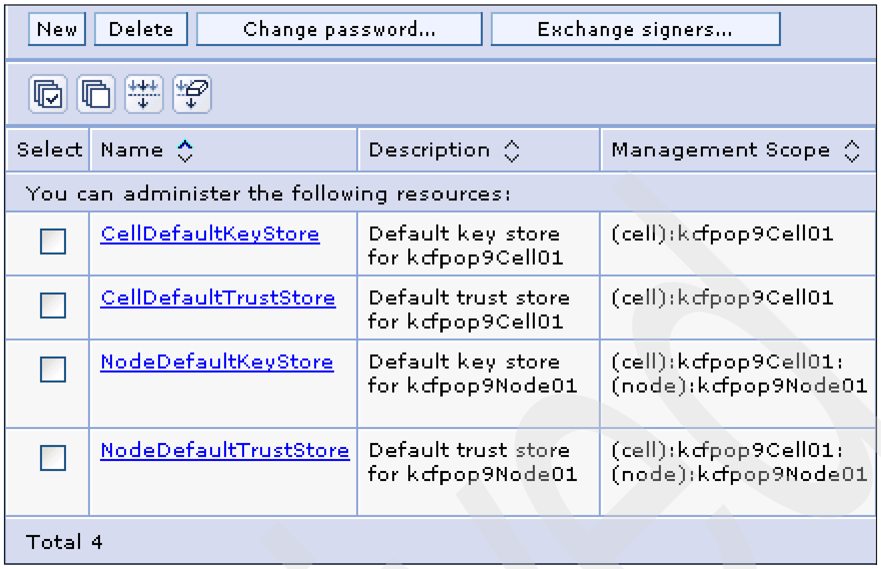
WebSphere Application Server supports following keystore types:

- JKS: Java KeyStore (`*.jks`)
- JCEKS: Java Cryptography Extension KeyStore (`*.jceks`)
- PKCS12: Public-Key Cryptography Standards #12 (*.p12), Microsoft® calls it PFX.
- PKCS12JarSigner
- PKCS11: Cryptographic Token Device
- CMSKS: Format used by IBM HTTP Server (`*.kdb`)
- JCERACFKS: z/OS only, stores certificates in RACF
- JCECCARACFKS: z/OS only, uses hardware cryptography device
- JCECCAKS: z/OS only, uses hardware cryptography device

Default keystores and truststores

During profile creation, a default keystore file (called `key.p12`) and a default truststore file (called `trust.p12`) are created. In addition, a default chained certificate is created in the `key.p12` file. The root signer, or public key, of the chained certificate is extracted from the `key.p12` file and added to the `trust.p12` file. If the files do not exist already, they are created during process startup.

The default keystores and truststores for the cell and each node can be easily identified by their suffixes: DefaultKeyStore and DefaultTrustStore. You can see the key and truststores in the administrative console by selecting **Security** → **SSL certificate and key management** → **keystores and certificates**. Select **SSL keystores** in the Keystore usage drop-down list (Figure 5-2 on page 156).



Select	Name	Description	Management Scope
You can administer the following resources:			
☐	CellDefaultKeyStore	Default key store for kcfpop9Cell01	(cell):kcfpop9Cell01
☐	CellDefaultTrustStore	Default trust store for kcfpop9Cell01	(cell):kcfpop9Cell01
☐	NodeDefaultKeyStore	Default key store for kcfpop9Node01	(cell):kcfpop9Cell01:(node):kcfpop9Node01
☐	NodeDefaultTrustStore	Default trust store for kcfpop9Node01	(cell):kcfpop9Cell01:(node):kcfpop9Node01
Total 4			

Figure 5-2 Default KeyStore and default TrustStore

The default keystores and truststores are located in the following locations:

- The default cell keystore and truststore are stored in the cell directory of the configuration repository:
  - *profile_root*/config/cells/*cell_name*/key.p12
  - *profile_root*/config/cells/*cell_name*/trust.p12
- The default node key and truststores are stored in the node directory of the configuration repository:
  - *profile_root*/config/cells/*cell_name*/nodes/*node_name*/key.p12
  - *profile_root*/config/cells/*cell_name*/nodes/*node_name*/trust.p12

Note: The default password is `WebAS` for all of the default keystores that are generated by WebSphere Application Server. You must change this password during profile creation for a more secure environment.

5.1.3 SSL configurations

WebSphere Application Server allows you to create multiple SSL configurations for use in specific situations. Each configuration points to a keystore and truststore to be used for connections using that configuration. SSL configurations are defined at specific management scopes. The scope that an SSL configuration inherits depends upon whether you create it using a cell, node, server, or endpoint link in the configuration topology.

When you create an SSL configuration, you can set the following SSL connection attributes:

- Keystore
- Default client certificate for outbound connections
- Default server certificate for inbound connections
- Truststore
- Key manager for selecting a certificate
- Trust manager or managers for establishing trust during the handshake
- Handshaking protocol
- Ciphers for negotiating the handshake
- Client authentication support and requirements

SSL configuration selection policy

There can be many SSL configurations and associated keystores and truststores in the cell. The SSL configuration to use at run time is selected in this order:

1. Programmatic selection

   Properties for an outbound connection can be specified on the thread by an application that directly uses the IBM JSSEHelperAPI. Subsequent implicit usage of JSSE (for example, opening an HTTPS connection with a URL provider) uses the thread-specific settings set by the JSSEHelperAPI. Refer to the following article for more details:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_ssloutconfiguseJSSE.html

2. Dynamic selection

   You can associate an SSL configuration dynamically with a specific outbound protocol, target host, and port, or by using a predefined selection criteria. When WebSphere establishes the connection, it checks to see if the target host and port match a predefined criteria, which applies for both application code usage and WebSphere Application Server usage. Refer to the following page for more details:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_ssldynoutboundconfigs.html

3. Direct selection

   An SSL configuration can be directly specified for this endpoint. You can make this specification administratively for WebSphere Application Server

SSL usage. Application programs can also reference an SSL configuration directly using the JSSEHelperAPI.

4. Scoped selection

   If none of the previous SSL configurations apply, the topology-based SSL configuration is selected.

5.2 Basic usage scenarios

SSL in WebSphere Application Server is mainly used for two purposes: to secure data during transfer (encryption) and to restrict access to certain services (authentication and authorization).

5.2.1 Securing administrative communication

Sensitive data can be sent over the network during administrative operations (for example, the administrator user name and password while executing wsadmin scripts). Fortunately, when you enable administrative security, SSL is automatically configured for all administrative transports (HTTP, SOAP, and so forth). You do not have to change this configuration unless you have custom requirements.

5.2.2 Securing LDAP communication

LDAP is the typical choice for user registry. However, by default, connection to the LDAP server is not protected in any way. While user data is sent using this connection, we recommend that you use secured and encrypted transport: SSL. Using SSL is extremely important when WebSphere Application Server has “write” access to LDAP and can modify the data stored there, such as creating users or changing a user’s group assignment.

The following steps describe how to configure SSL:

1. Enable SSL on your LDAP server. This step is product-specific and is usually performed by the LDAP administrator.
2. Obtain the public certificate for the LDAP server from the LDAP administrator. This certificate will be added in a later step to the truststore of the WebSphere Application Server to establish a connection to LDAP. If the LDAP server’s certificate is signed by a Certificate Authority (CA), you can use a signer certificate instead.

3. Create a keystore at the cell scope, which will ensure that all servers have access to the keystore. Refer to 5.3.1, “Creating keystores” on page 163 for more details.

   Best practice: We recommend that you create a separate keystore and SSL configuration for the LDAP connection, which ensures that the LDAP server certificate or signing certificate (if used) is only trusted for LDAP connections and not other connections.

4. Import the LDAP certificate into the keystore as a signer certificate. Refer to 5.3.3, “Managing signer certificates” on page 169.
5. Create a new SSL configuration at the cell scope. Select the new keystore as both the keystore and truststore. Refer to 5.3.7, “Creating SSL configurations” on page 177.
6. Enable SSL in the user registry configuration.

   On the user registry configuration page (for a stand-alone LDAP registry, or LDAP repository that is part of the federated repositories), enable SSL, choose **Use specific SSL alias**, and select the created SSL configuration.

   Best practice: We recommend that you use a specific alias rather than a centrally managed configuration in this case to ensure that all servers will use this configuration to connect to LDAP.

Now, you have a secure connection to LDAP, and user data cannot be stolen during transmission.

For detailed instructions about securing an LDAP connection, refer to 5.6.1, “Securing LDAP communication” on page 193.

5.2.3 Securing Web inbound and outbound communication

For Web communication, there are two use cases:

- You want to secure inbound traffic to the server.
- You want to call an external application over a secured connection.

In either case, there might be an additional requirement of client certificate authentication.

Inbound communication

Most Web applications transmit sensitive data, for example, a user name and password during login or personal data during the interaction with the application. To make this data safe during transfer, we use SSL.

In the WebSphere environment, we recommend that you access application servers through a Web server, for example, IBM HTTP Server (IHS).

If client certificate authentication is not required, perform the following steps to configure SSL communication:

1. Configure the Web server for SSL (refer to “Configuring the Web server for SSL” on page 203):
   a. Create the key database file and certificates required for the Web server to participate in an SSL connection. The certificate must be signed by a well known CA.
   b. Enable the directives in the Web server configuration for SSL, pointing to the new key database.

   This step allows SSL connections to be established between Web browsers and the Web server.
2. Configure the HTTP Plug-in for SSL connections (refer to “Configuring the plug-in for SSL connection” on page 205):
   a. Add the Web server definition to WebSphere (which is usually done as a part of the HTTP plug-in configuration process).

      When a Web server definition is created, it is associated with a keystore that contains all of the signers for the cell and the chained certificate for the Web server node.
   b. Copy the Web server keystore and stash files for the plug-in to the Web server plug-in location.

For detailed instructions about securing a Web connection, refer to 5.6.2, “Securing Web inbound communication” on page 203.

If client certificate authentication is required, configuration is more complex. In addition to the previous steps, you have to configure the Web server to require client certificates and configure mutual trust between the plug-in and the application server. For more information, refer to Chapter 8, “Securing a Web application” on page 267.

Outbound communication

Applications might need to communicate with external services. These external services usually require encryption and often certificate authentication also.

We recommend that you create separate SSL configurations for each external service to provide flexibility and isolation. Depending on your requirements, the number of external services, and the topology, you can select a specific SSL configuration selection method. Refer to “SSL configuration selection policy” on page 157.

The following steps describe how to prepare SSL configuration for external service:

1. Create a keystore at the appropriate scope. Choose a scope that will allow access to the keystore for all servers that have to connect to the external service. Refer to 5.3.1, “Creating keystores” on page 163.
2. Obtain the certificate from the external service server.
3. Import the certificate into the keystore as a signer certificate. Refer to 5.3.3, “Managing signer certificates” on page 169.
4. If client certificate authentication is required:
   a. If the service provider provides you with a client certificate, import it as a personal certificate into the keystore.
   b. Otherwise:
      i. Generate a new self-signed personal certificate or chained certificate.
      ii. Extract the public part of the certificate or root signer certificate.
      iii. Send the extracted certificate to the service provider where it must be added as a trusted certificate to allow a connection to be established.

   Refer to 5.3.2, “Managing personal certificates” on page 165 for task details.
5. Create a new SSL configuration at the same scope. Select the new keystore as both the keystore and the truststore. Refer to 5.3.7, “Creating SSL configurations” on page 177.
6. Ensure that the SSL configuration will be used. Refer to “SSL configuration selection policy” on page 157.

5.2.4 Securing EJB inbound and outbound communication

When you enable security, Remote Method Invocation (RMI)/Internet Inter-ORB Protocol (IIOP) traffic is automatically protected by default SSL configuration. You can customize this configuration in a way similar to Web communication. For more details, refer to Chapter 6, “Common Secure Interoperability Version 2 administration” on page 209.

5.2.5 Securing communication with WebSphere MQ

Many existing systems use WebSphere MQ for integration. By default, WebSphere MQ communications are unsecured. If WebSphere MQ is used as your Java Message Service (JMS) provider, we recommend that you establish mutual SSL communication between WebSphere Application Server and WebSphere MQ.

The following steps describe how to configure SSL to WebSphere MQ:

1. Enable SSL on WebSphere MQ; refer to WebSphere MQ-related publications for details.
2. Obtain the public certificate for WebSphere MQ from the WebSphere MQ administrator.
3. Create a keystore at the appropriate scope. Choose a scope that will allow all servers that have to connect to WebSphere MQ to have access to the keystore. Refer to 5.3.1, “Creating keystores” on page 163.
4. Import the obtained certificate into the keystore as a signer certificate. Refer to 5.3.3, “Managing signer certificates” on page 169.
5. In the keystore, create a new personal certificate for authentication to WebSphere MQ. Refer to “Creating a chained personal certificate” on page 167.
6. Extract the public part of the new certificate and provide it to the WebSphere MQ administrator, because it must be added to the MQ truststore. Refer to 5.3.2, “Managing personal certificates” on page 165.
7. Create a new SSL configuration at the same scope. Select the new keystore as the keystore and truststore. Refer to 5.3.7, “Creating SSL configurations” on page 177.
8. Configure the JMS MQ connection factory in WebSphere:
   a. Set the correct Server connection channel.
   b. Enable SSL and select the new SSL configuration.
   c. Additionally, in the Client transport properties, you can set the peer name that will be checked against the distinguished name (DN) of the MQ certificate.

Now, if WebSphere MQ is configured correctly, it will only allow connections using SSL from parties that have the correct certificate.

5.3 Basic SSL administration

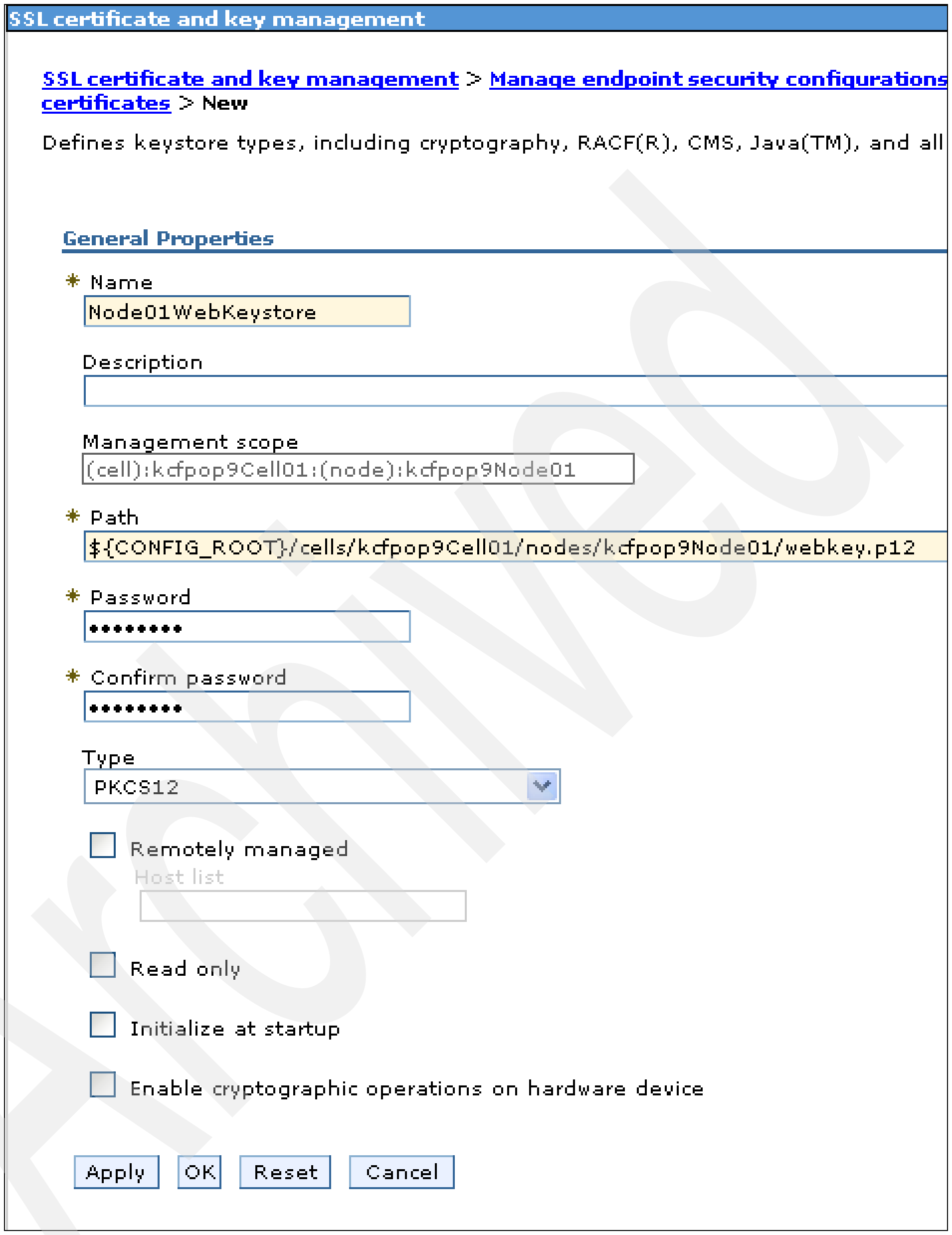
This section discusses WebSphere administration tasks that are associated with SSL management.

5.3.1 Creating keystores

You need a keystore to hold certificates. To connect over SSL to any external party, you have to trust that party; you have to add that party's signer certificate to the application server truststore. Creating additional keystores for this purpose increases the security of the environment, because a certificate is trusted only for certain communications, and not globally.

To configure a new keystore, perform the following steps:

1. In the administrative console, click **Security** → **SSL certificate and key management** → **Manage endpoint security configurations** → **{Inbound | Outbound}**, and select the scope for your new keystore as cell, node group, cluster, node, server, or endpoint. For keystore configuration, inbound or outbound topology does not matter, the new keystore is available for both inbound and outbound in that scope after the new keystore is created.
2. Under **Related Items**, click **keystores and certificates**, and then, click **New** (Figure 5-3 on page 164).



SSL certificate and key management

SSL certificate and key management > Manage endpoint security configurations
certificates > New

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all

General Properties

* Name
Node01WebKeystore

Description

Management scope
(cell):kcfpop9Cell01:(node):kcfpop9Node01

* Path
${CONFIG_ROOT}/cells/kcfpop9Cell01/nodes/kcfpop9Node01/webkey.p12

* Password
•••••••••

* Confirm password
•••••••••

Type
PKCS12

☐ Remotely managed
Host list

☐ Read only

☐ Initialize at startup

☐ Enable cryptographic operations on hardware device

Figure 5-3 Creating a new SSL Keystore

To create a new SSL keystore:

– Type a name in the Name field. This name uniquely identifies the keystore in the configuration.

   - Type the location of the keystore file in the Path field. The location can be a file name or a file URL to an existing keystore file. We recommend that you use WebSphere environment variables in this field.
   - Type the keystore password in the Password field. This password is for the keystore file that you specified in the Path field.
   - Type the keystore password again in the Confirm Password field to confirm the password.
   - Select a keystore type from the list. The type that you select is for the keystore file that you specified in the Path field. The recommended keystore type is `PKCS12`.
   - Select any of the following optional selections:
     - The Read only selection creates a keystore configuration object, but it does not create a keystore file. If this option is selected, the keystore file that you specified in the Path field must already exist.
     - The "Initialize at startup" selection initializes the keystore during run time.
     - The "Enable cryptographic operations on a hardware device" option specifies whether a hardware cryptographic device is used for cryptographic operations only. Operations that require login are not supported when using this option.
3. Click **Apply** and **Save** to save the configuration.

5.3.2 Managing personal certificates

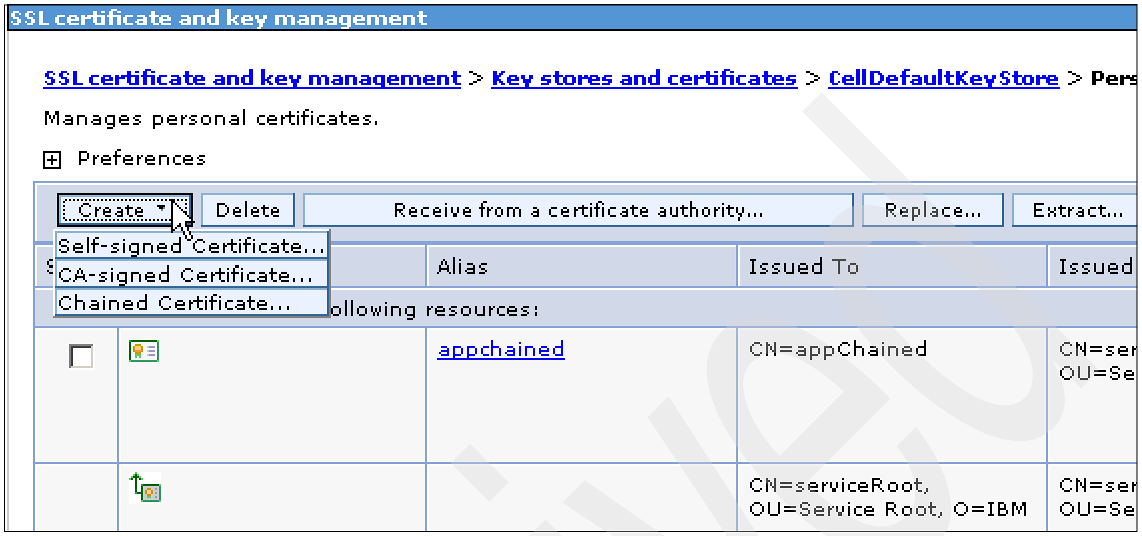
Personal certificates can be used as server certificates and as client certificates. When used as server certificates, they allow an external client to verify the identity of the server. When used as client certificates, they provide a client identity to the external server.

Most often, a personal certificate is created when you want to enable HTTPS on the Web server or if you need to authenticate a client, for example, when setting up mutual trust between the HTTP plug-in and the Web container or between an application server and WebSphere MQ.

To manage personal certificates:

1. Open the administrative console, and click **Security** → **SSL certificate and key management**.
2. On the right side of the panel under Related Items, click **keystores and certificates**.
3. Click a keystore name to which you want to add the personal certificate.

4. Under Additional Properties, click **Personal certificates**. The Personal certificates management page is opened as shown in Figure 5-4.


SSL certificate and key management

SSL certificate and key management > Key stores and certificates > CellDefaultKeyStore > Pers

Manages personal certificates.

⊞ Preferences

Create ▾	Delete	Receive from a certificate authority...	Replace...	Extract...

Self-signed Certificate...
CA-signed Certificate...
Chained Certificate...

S		Alias	Issued To	Issued
ollowing resources:				
☐		appchained	CN=appChained	CN=ser OU=Se
			CN=serviceRoot, OU=Service Root, O=IBM	CN=ser OU=Se

Figure 5-4 Personal certificates management page

You can perform the following tasks on this page:

- Create personal certificates. You can create the following certificates:
  - Self-signed Certificate: The self-signed certificate is the most basic type. Because it is self-signed, it is best used in an internal or controlled environment. The self-signed certificate can be used in communication with business partners; however, renewing the certificate implies changes on both sides.
  - Chained Certificates: A chained certificate is a personal certificate that is signed by a root certificate. Using a chained certificate enables you to refresh the personal certificate without affecting the trust established.
  - CA-signed Certificates: A CA-signed certificate is a personal certificate that is signed by a Certificate Authority. It is best used for public endpoints, because most clients or browsers already have signer certificates from trusted authorities and can validate a certificate without a signer exchange.

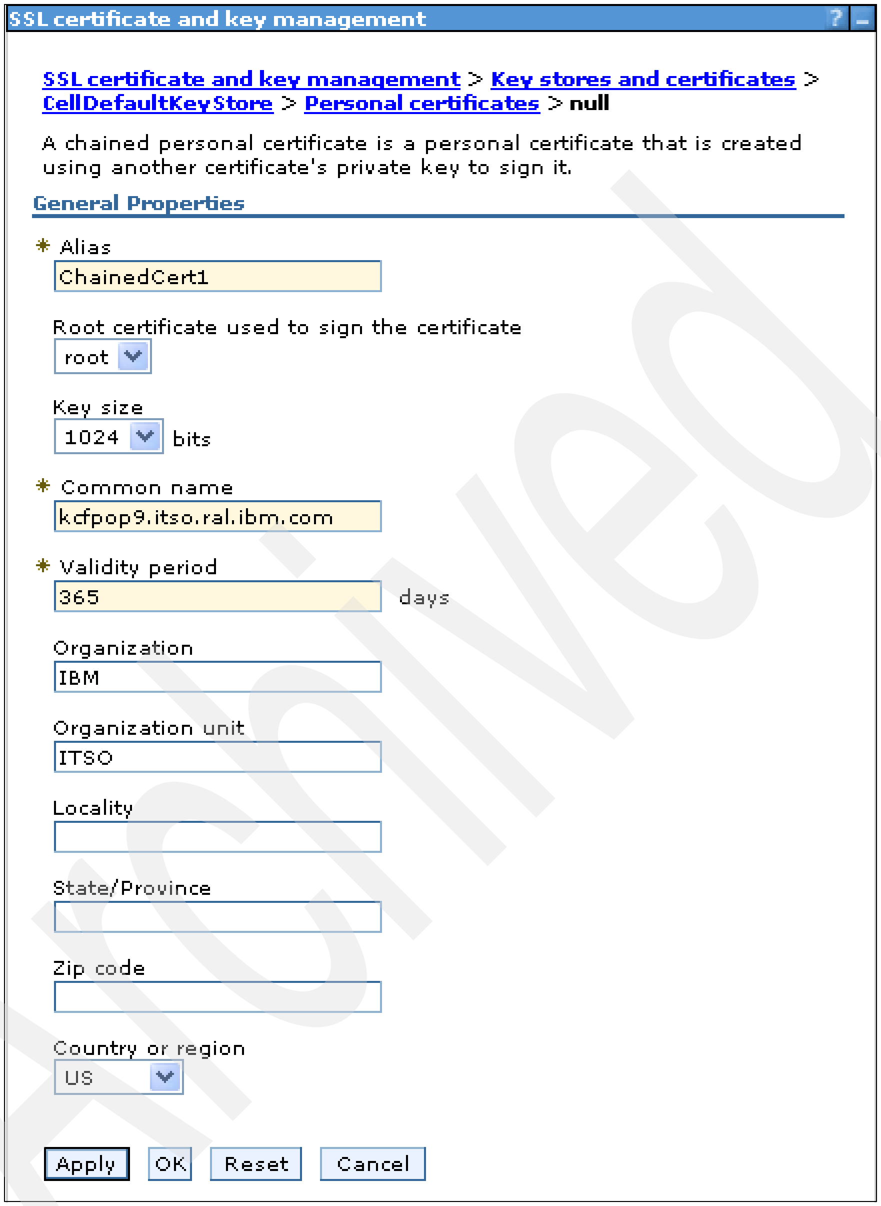
    The CA client must be configured to create CA-signed certificates. Refer to 5.4.2, “Creating and defining a CA client” on page 186.
- Delete a personal certificate: You can delete a certificate, when you no longer need it. Deleted certificates are placed in the Deleted certificate keystore.

- Receive certificate from a CA: This option is used to get CA-signed certificates without a CA client. First, you need to create a certificate request via the Personal certificate requests page and send it to the CA. When you receive the response, you can integrate it with the request via this function.
- Replace certificate: The first selected certificate is replaced by the second selected certificate. All places where the old alias was used are updated with the new alias. You can optionally delete the old certificate and signer.
- Extract: This option extracts the public part of the personal certificate. The extracted certificate is added to the other party's truststore to establish the connection.
- Import: This option imports the personal certificate from a file or keystore, for example, from a p12 file. You can use this function to import a personal certificate that was created using other software.
- Export: You can export a personal certificate (with private keys) to a file or a keystore. You can export a certificate for usage in other software or for backup purposes.
- Revoke: This option revokes certificates that are signed by a CA. Use this function when a certificate has been compromised.
- Renew: This option recreates the certificate with all the information from the original certificate, but with a new expiration period and public/private key pair. Only self-signed certificates and chained certificates that were created by WebSphere can be renewed.

Creating a chained personal certificate

The server's default personal certificate is a chained certificate that is created when the profile is created. A new chained personal certificate can be created through the administrative console after profile creation using the following steps:

1. Log in to the administrative console, and click **Security** → **SSL certificate and key management**.
2. On the right side of the panel under Related Items, click **keystores and certificates**.
3. Click a keystore name to which you want to add the chained personal certificate.
4. Under Additional Properties, click **Personal certificates**.
5. Click **Create**, and select **Chained Certificate**.
6. Fill in the information in the General Properties section (Figure 5-5 on page 168), and select **Root certificate** from the pull-down list.


SSL certificate and key management

SSL certificate and key management > Key stores and certificates > CellDefaultKeyStore > Personal certificates > null

A chained personal certificate is a personal certificate that is created using another certificate's private key to sign it.

General Properties

* Alias
ChainedCert1

Root certificate used to sign the certificate
root

Key size
1024 bits

* Common name
kcfpop9.itso.ral.ibm.com

* Validity period
365 days

Organization
IBM

Organization unit
ITSO

Locality

State/Province

Zip code

Country or region
US

Figure 5-5 General properties of personal certificate

7. Click **Apply**, click **OK**, and then, click **Save** to save the configuration.

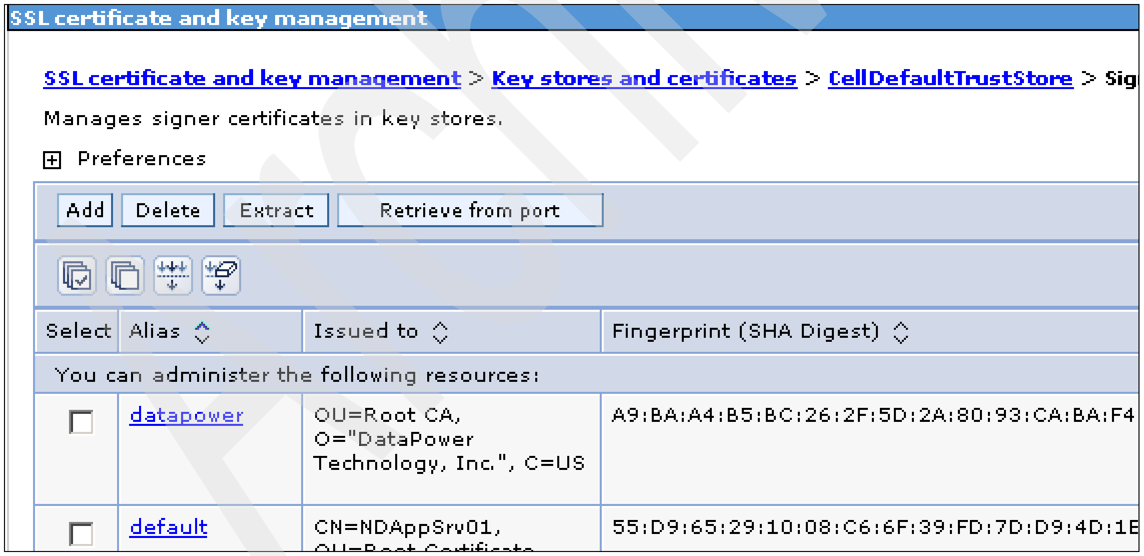
After saving the configuration, a new chained personal certificate, which is signed by the root, is created and stored in the keystore. This certificate can be used by the run time for SSL communication.

5.3.3 Managing signer certificates

A peer signer certificate must be trusted (added to the truststore) to successfully establish an SSL connection. Whenever you want to establish an SSL connection to an external service, for example, an LDAP server, you must have its signer certificate in the truststore.

To manage signer certificates:

1. Open the administrative console, and click **Security** → **SSL certificate and key management**.
2. On the right side of the panel under Related Items, click **keystores and certificates**.
3. Click a keystore name to which you want to add the signer certificate.
4. Under Additional Properties, click **Signer certificates**.
5. The Signer certificates management page is opened as shown in Figure 5-6.



SSL certificate and key management

SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Sig

Manages signer certificates in key stores.

⊞ Preferences

Add | Delete | Extract | Retrieve from port

Select	Alias	Issued to	Fingerprint (SHA Digest)
You can administer the following resources:			
☐	datapower	OU=Root CA, O="DataPower Technology, Inc.", C=US	A9:BA:A4:B5:BC:26:2F:5D:2A:80:93:CA:BA:F4
☐	default	CN=NDAppSrv01,	55:D9:65:29:10:08:C6:6F:39:FD:7D:D9:4D:1E

Figure 5-6 Signer certificates management page

You can perform the following tasks on this page:

- Add certificate: Allows you to add the signer certificate from a file.

- Delete certificate: The option deletes the signer certificate. It removes any signer certificates that are no longer needed.
- Extract: This option extracts a signer certificate to a file.
- Retrieve from port: This option makes a test connection to an SSL port and retrieves the signer from the server during the handshake. It does not retrieve the root certificate if the server's certificate is a chained certificate.
- Exchange signer (available on the keystores page): This option allows you to exchange signers between keystores. It is extremely useful when configuring mutual trust between a plug-in and an application server to exchange their signer certificates.

5.3.4 Recovering deleted certificates

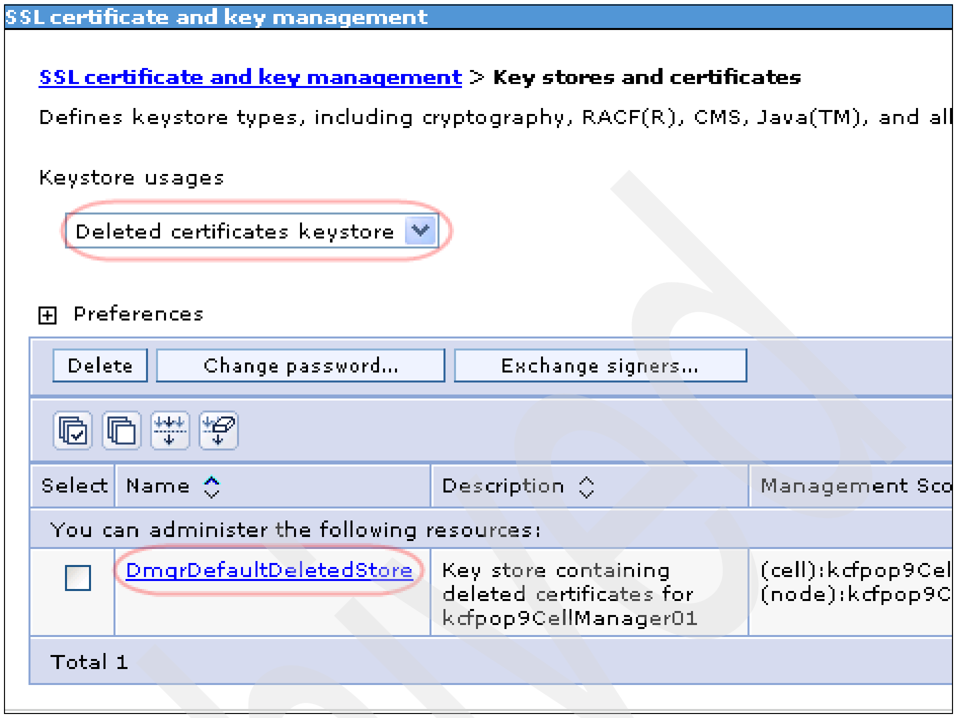
In WebSphere Application Server V7, there is a new enhancement for certificate deletion. The SSL configuration contains a recovery keystore to hold personal certificates that were deleted. There are separate recovery keystores for a deployment manager and an application server. On a stand-alone application server, the keystore is called `NodeDefaultDeletedStore`. On a deployment manager, the keystore is called `DmgrDefaultDeletedStore`.

When a personal certificate is deleted from a keystore using the administrative console or in a script using the `deleteCertificate` AdminTask, a copy of the certificate is stored in DmgrDeletedKeyStore or NodeDeletedKeyStore. You have the option to recover that certificate by using the `importCertificate` or `exportCertificate` tasks.

If a personal certificate is deleted from the recovery keystore, it is permanently deleted and cannot be recovered. The recovery keystore will be emptied when the certificate expiration monitor is run, and it can insure that the recovery keystore does not hold certificates forever.

To recover a personal certificate using the administrative console, perform the following steps:

1. Click **Security** → **SSL certificate and key management**.
2. Under Related Items, click **keystores and certificates**.
3. From the Keystore usages drop-down list at the top of the page, select **Deleted certificates keystore** as shown in Figure 5-7 on page 171.


SSL certificate and key management

SSL certificate and key management > Key stores and certificates

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all

Keystore usages

Deleted certificates keystore

Preferences

Select	Name	Description	Management Sco
You can administer the following resources:			
☐	DmgrDefaultDeletedStore	Key store containing deleted certificates for kcfpop9CellManager01	(cell):kcfpop9Cel (node):kcfpop9C
Total 1			

Figure 5-7   Deleted certificates keystore

4. Click **DmgrDefaultDeletedStore** or **NodeDefaultDeletedStore** depending on the environment difference.
5. Under Additional Properties, click **Personal certificates**.
6. Select a certificate.
7. Select **Export**.
8. Perform these steps:
   – Enter the keystore password of the deleted keystore.
   – Enter the alias to be assigned to the certificate (in the keystore that will receive the certificate).
   – Select **Managed keystore**.
   – Select the keystore from the drop-down list that will receive the certificate.
   – Click **Apply**, and then, click **OK**.

5.3.5 Certificate expiration monitoring

Certificates are created with a finite life span. When certificates expire, they can no longer be used by the system and can cause system outages. WebSphere Application Server provides a utility to monitor certificates that are close to expiration or have already expired. This utility is a scheduled task that cycles through all the keystores in the security configuration, and it will report if any certificates are expired, fall within the expiration threshold, or fall within the pre-notification period. It also can be configured to delete the expired certificates and recreate certificates.

Certificate expiration monitoring relies on the following definitions:

- Expired certificates

  If certificates have reached the end of their life span, self-signed or chained certificates are reported and replaced based on the configuration setting. CA-signed certificates cannot be replaced but will be reported.

- Certificates within the expiration threshold

  The server replaces certificates to ensure that the certificate does not expire if the certificate falls within the expiration threshold.

- Pre-notification period

  The *pre-notification period* is the period of time prior to the expiration threshold date, and when a certificate falls within this threshold, there are warnings issued that indicate that the certificate will be replaced.

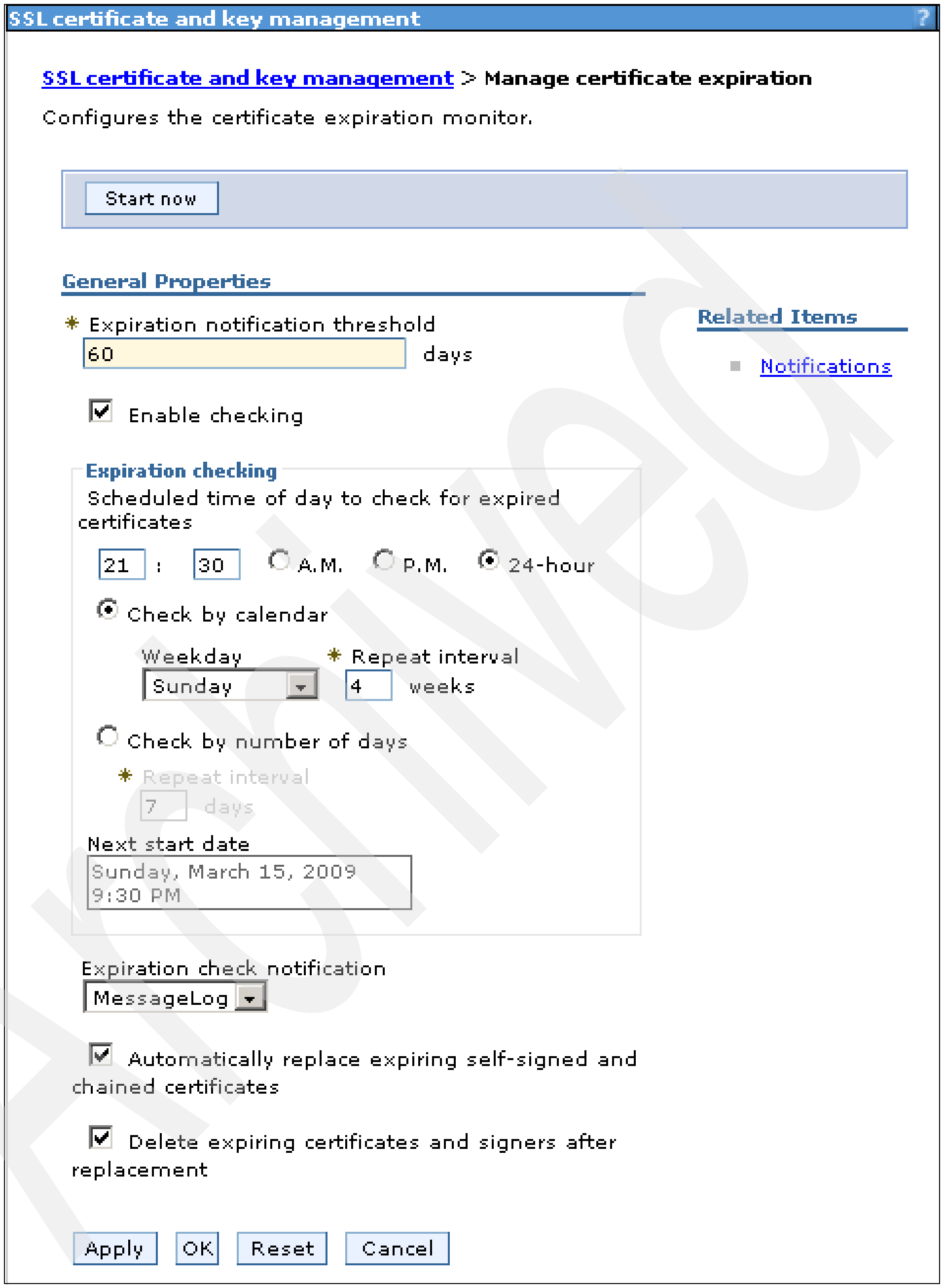
The certificate expiration monitor performs these steps:

1. The certificate expiration monitor clears out the NodeDefaultDeletedStore or DmgrDefaultDeletedStore. This operation is performed silently without reporting that the certificates are deleted.
2. The certificate expiration monitor checks the root keystores, DmgrDefaultRootStore or NodeDefaultRootStore, and the DmgrRSATokenRootStore or NodeRSATokenRootStore. If any root certificates are expired or fall in the threshold period or the pre-notification period, the certificate is noted in the report.
3. If there are any root certificates that are expired or fall in the threshold period, that root certificate is recreated using all the information that was used to create the original root certificate. Any signer certificates from the original root certificate are replaced with the signers from the new root certificate.
4. If a root certificate is replaced, all the keystores are checked to see if there are any chained certificates signed with the original root certificate. If there are any chained certificates signed with the original root certificate, the chain certificate is renewed (recreated with the new root certificate). Any signer

certificate from the original certificate is replaced with the signer from the recreated certificate.

5. After all root keystores are processed, the rest of the keystores are checked for expired certificates, certificates in the expiration threshold, or certificates in the pre-notification period. Any certificate falling in any one of these categories is noted in the report.
6. If there are any expired certificates or certificates in the expiration threshold period and these certificates are self-signed certificates or chained certificates created by WebSphere, they are replaced. If the chained certificate's root is not in the root keystore, it will be recreated as a default root certificate. Any signer certificates from the original certificate are replaced with the signer from the new certificate.
7. A report is generated and returned, written to a log file, or mailed.

Certificate expiration monitoring can be configured in the administrative console through clicking **Security** → **SSL certificate and key management** → **Manage certificate expiration** as shown in Figure 5-8 on page 174.


SSL certificate and key management

SSL certificate and key management > Manage certificate expiration

Configures the certificate expiration monitor.

General Properties

* Expiration notification threshold
60 days

[✓] Enable checking

Expiration checking

Scheduled time of day to check for expired certificates
21 : 30 ( ) A.M. ( ) P.M. [✓ 24-hour]

[✓ Check by calendar]

Weekday: Sunday
* Repeat interval: 4 weeks

( ) Check by number of days

* Repeat interval: 7 days

Next start date
Sunday, March 15, 2009
9:30 PM

Expiration check notification
MessageLog

[✓] Automatically replace expiring self-signed and chained certificates

[✓] Delete expiring certificates and signers after replacement

Related Items
- Notifications

Figure 5-8 Manage certificate expiration

You can configure the number of days threshold in the Expiration notification threshold field, enable certificate monitoring by selecting **Enable checking**, enable **Automatically replace expiring self-signed certificates**, enable **Delete expiring certificates and signers after replacement**, select the method of notification, and schedule running the certificate expiration monitor. You also can start the certificate expiration monitor any time by clicking **Start now**.

By default, the expiration threshold is 60 days and the pre-notification period is 90 days. The expiration threshold can be configured as required on the panel that is shown in Figure 5-8 on page 174. The pre-notification period is defined in the `com.ibm.ws.security.expirationMonitorNotificationPeriod` property.

The expiration monitor automatically replaces only self-signed certificates and chained certificates that are expired or that meet the expiration threshold criteria. Self-signed certificates are renewed using all the information that was used to create the original self-signed certificate. A chained certificate is renewed using the same root certificate that was used to sign the original certificate.

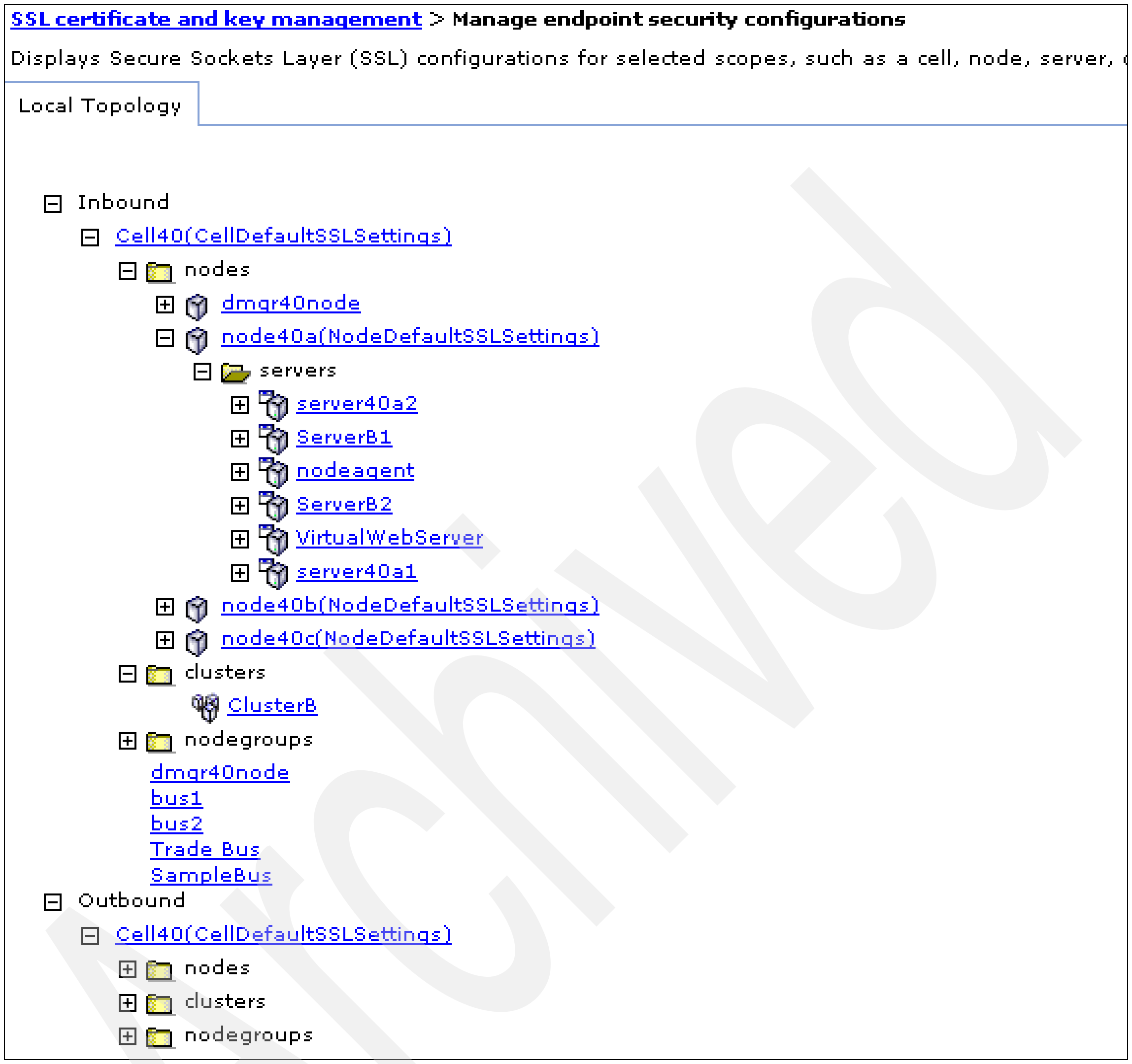
Note: When the expiration monitor replaces certificates, the run time is affected when “SSL Dynamically update” is enabled.

5.3.6 Managing SSL configurations

SSL configurations are managed from a central location in the administrative console (Figure 5-9 on page 176). You can associate an SSL configuration and certificate with a specific management scope, which allows you to make changes for the entire topology using the cell scope and specific changes using a particular endpoint name for a specific application server process. Configuration settings at a higher level scope are inherited by configurations at a lower scope.

The configuration scoping is separated into inbound and outbound management scopes that represent opposing directions during the connection handshake process.

The topology view that provides the central management capability for SSL configurations can be found in the administrative console by selecting **Security → SSL certificates and key management → Manage endpoint security configurations** (Figure 5-9 on page 176).



SSL certificate and key management > Manage endpoint security configurations

Displays Secure Sockets Layer (SSL) configurations for selected scopes, such as a cell, node, server,

Local Topology

```
⊟ Inbound
    ⊟ Cell40(CellDefaultSSLSettings)
        ⊟ nodes
            ⊞ dmgr40node
            ⊟ node40a(NodeDefaultSSLSettings)
                ⊟ servers
                    ⊞ server40a2
                    ⊞ ServerB1
                    ⊞ nodeagent
                    ⊞ ServerB2
                    ⊞ VirtualWebServer
                    ⊞ server40a1
            ⊞ node40b(NodeDefaultSSLSettings)
            ⊞ node40c(NodeDefaultSSLSettings)
        ⊟ clusters
              ClusterB
        ⊞ nodegroups
          dmgr40node
          bus1
          bus2
          Trade Bus
          SampleBus
⊟ Outbound
    ⊟ Cell40(CellDefaultSSLSettings)
        ⊞ nodes
        ⊞ clusters
        ⊞ nodegroups
```

Figure 5-9 Central SSL management

The SSL configuration scope encompasses the level where you created the configuration and all the levels below that point. For example, when you create an SSL configuration at a specific node, that configuration can be seen by that node agent and by every application server that is part of that node. Any application server or node that is not part of this particular node cannot see this SSL configuration.

In order from highest (least specific) to lowest, the scopes are:

- Cell
- Node group
- Node, server
- Cluster
- Endpoint

5.3.7 Creating SSL configurations

SSL configurations contain the attributes that you need to control the behavior of client and server SSL endpoints. While central management makes it easy to utilize a single SSL configuration for securing an entire cell, we recommend that you create separate SSL configurations within specific management scopes on the inbound and outbound tree in the configuration topology. You can, however, utilize a single keystore for managing certificates that coincide with different SSL configurations for securing various transports. You can secure the following transports with an SSL configuration:

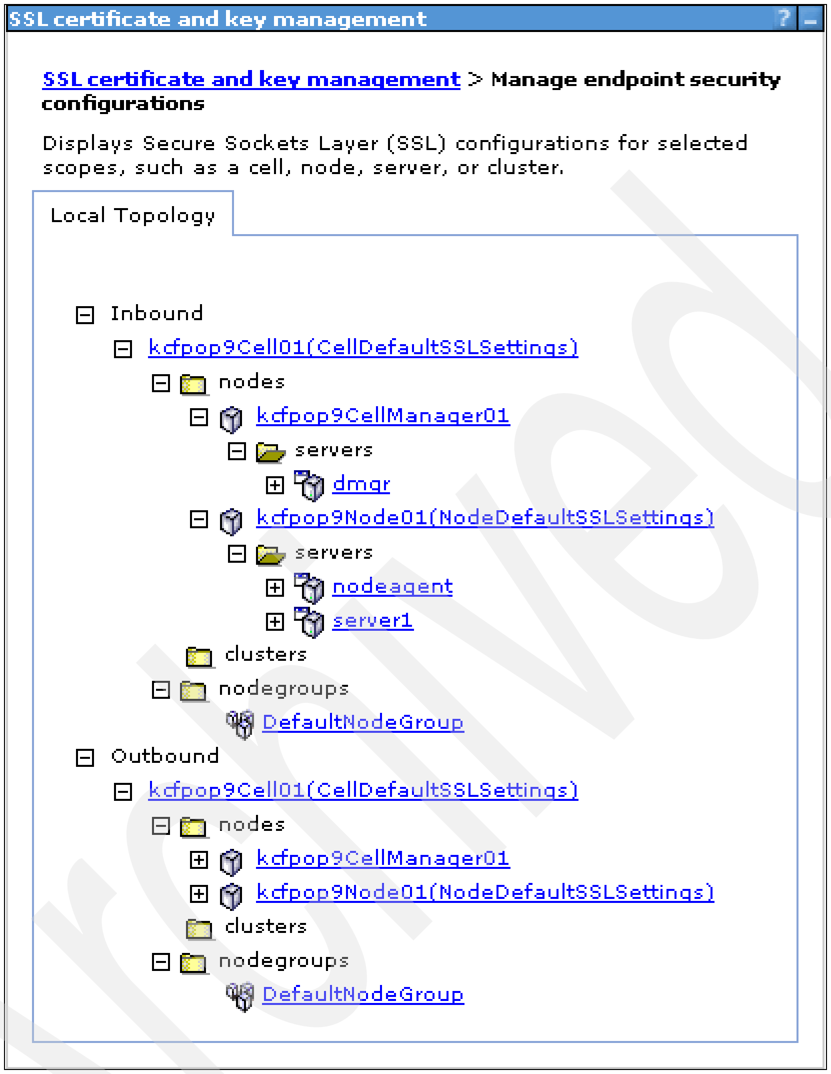
- Hypertext Transfer Protocol (HTTP)
- Lightweight Directory Access Protocol (LDAP)
- Internet InterORB Protocol (IIOP)
- Simple Object Access Protocol (SOAP)
- Session Initiation Protocol (SIP)
- Service Integration Bus (SIB)
- WebSphere MQ connection (MQ)

With the integration of keystore and certificate management, specific certificates from a keystore can be associated with an SSL configuration.

Perform the following steps to create the new SSL configuration:

1. Click **Security** → **SSL certificate and key management** → **Manage endpoint security configurations**.
2. Select an SSL configuration link on either the Inbound or Outbound tree, depending on the process that you configure.

   If the scope is already associated with a configuration and alias, the SSL configuration alias and certificate alias are noted in parentheses. Otherwise, the scope is not associated, as shown in Figure 5-10 on page 178. Instead, the scope inherits the configuration properties of the first scope above it that is associated with an SSL configuration and certificate alias. The cell scope must be associated with an SSL configuration, because it is at the top of the topology and represents the default SSL configuration for the inbound or outbound connection.


SSL certificate and key management

SSL certificate and key management > Manage endpoint security configurations

Displays Secure Sockets Layer (SSL) configurations for selected scopes, such as a cell, node, server, or cluster.

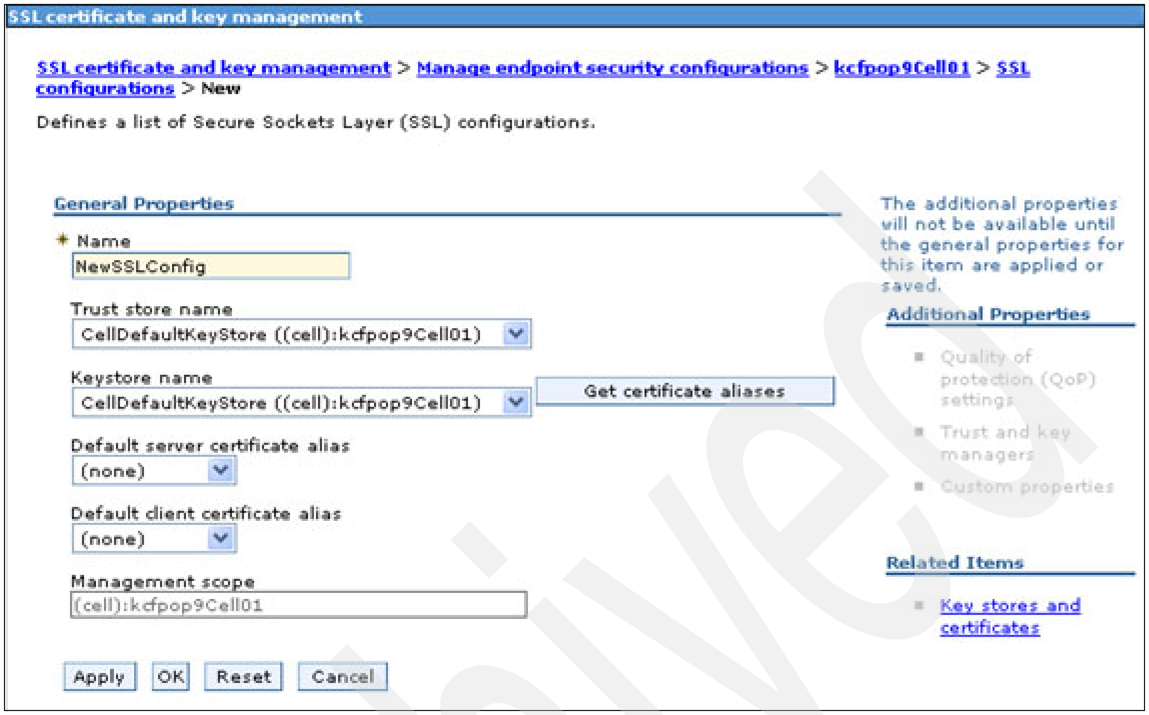
Local Topology

```
⊟ Inbound
    ⊟ kcfpop9Cell01(CellDefaultSSLSettings)
        ⊟ nodes
            ⊟ kcfpop9CellManager01
                ⊟ servers
                    ⊞ dmgr
            ⊟ kcfpop9Node01(NodeDefaultSSLSettings)
                ⊟ servers
                    ⊞ nodeagent
                    ⊞ server1
            clusters
        ⊟ nodegroups
              DefaultNodeGroup
⊟ Outbound
    ⊟ kcfpop9Cell01(CellDefaultSSLSettings)
        ⊟ nodes
            ⊞ kcfpop9CellManager01
            ⊞ kcfpop9Node01(NodeDefaultSSLSettings)
            clusters
        ⊟ nodegroups
              DefaultNodeGroup
```

Figure 5-10 SSL configuration scope

A configuration page for the selected endpoint will open.

3. Click **SSL configurations** under Related Items. You can view and select any of the SSL configurations that are configured at this scope or higher.
4. Click **New** to display the SSL configuration panel (Figure 5-11 on page 179).


SSL certificate and key management

SSL certificate and key management > Manage endpoint security configurations > kcfpop9Cell01 > SSL configurations > New

Defines a list of Secure Sockets Layer (SSL) configurations.

General Properties

* Name
NewSSLConfig

Trust store name
CellDefaultKeyStore ((cell):kcfpop9Cell01)

Keystore name
CellDefaultKeyStore ((cell):kcfpop9Cell01)

Default server certificate alias
(none)

Default client certificate alias
(none)

Management scope
(cell):kcfpop9Cell01

The additional properties will not be available until the general properties for this item are applied or saved.

Additional Properties
- Quality of protection (QoP) settings
- Trust and key managers
- Custom properties

Related Items
- Key stores and certificates

Figure 5-11 New SSL configuration

Enter the general properties for the new SSL configuration:

– Type an SSL configuration name.

– Select a trust store name from the drop-down list.

A *truststore name* refers to a specific truststore that holds signer certificates that validate the trust of certificates sent by remote connections during an SSL handshake.

– Select a keystore name from the drop-down list.

A *keystore* contains the personal certificates that represent a signer identity and the private key that WebSphere Application Server uses to encrypt and sign data.

If you change the keystore name, click **Get certificate aliases** to refresh the list of certificates from which you can choose a default alias. WebSphere Application Server uses a server alias for inbound connections and a client alias for outbound connections.

- Choose a default server certificate alias for inbound connections.

  Select the default only when you have not specified an SSL configuration alias elsewhere and have not selected a certificate alias. A centrally managed SSL configuration tree can override the default alias.
- Choose a default client certificate alias for outbound connections.

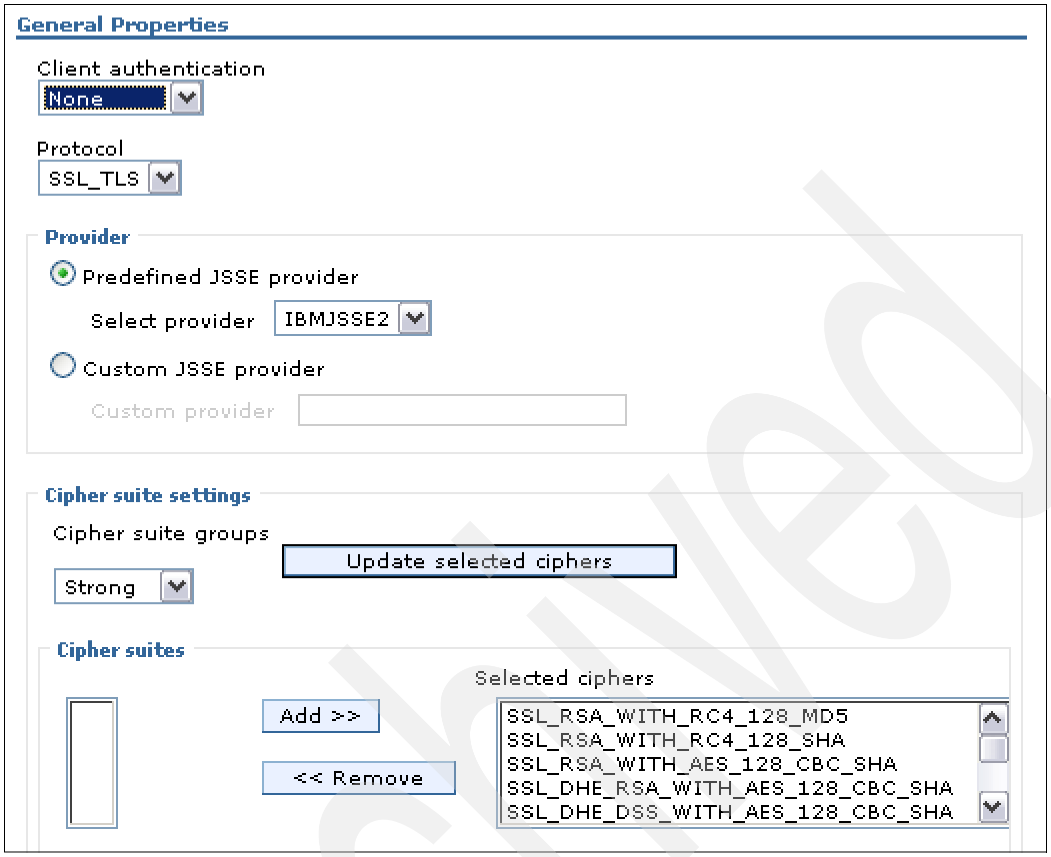
  Select the default only when the server SSL configuration specifies an SSL client authentication.

Click **Apply** to configure Additional Properties.

Quality of protection (QoP) settings

Quality of protection (QoP) settings define the strength of the SSL encryption, the integrity of the signer, and the authenticity of the certificate. You can select the client authentication setting, the protocol for the SSL handshake, and the Cipher suite settings.

The settings for this option are shown in Figure 5-12 on page 181.



General Properties

Client authentication: None

Protocol: SSL_TLS

Provider

Predefined JSSE provider [✓]

Select provider: IBMJSSE2

Custom JSSE provider

Custom provider

Cipher suite settings

Cipher suite groups: Strong

Update selected ciphers

Cipher suites

Add >>

<< Remove

Selected ciphers

```
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_AES_128_CBC_SHA
SSL_DHE_RSA_WITH_AES_128_CBC_SHA
SSL_DHE_DSS_WITH_AES_128_CBC_SHA
```

Figure 5-12 Quality of protection (QoP) settings

The client authentication setting is used to establish an SSL configuration for inbound connections and for clients to send their certificates:

- If you select None, the server does not request that a client send a certificate during the handshake.
- If you select Supported, the server requests that a client send a certificate. However, if the client does not have a certificate, the handshake might still succeed.
- If you select Required, the server requests that a client send a certificate. However, if the client does not have a certificate, the handshake fails.

Note: The signer certificate that represents the client must be in the truststore that you select for the SSL configuration. By default, servers within the same cell trust each other, because they use the common truststore, trust.p12, that is located in the cell directory of the configuration repository. However, if you use keystores and truststores that you create, perform a signer exchange before you select either Supported or Required.

The cipher suite settings specify the various cipher suite groups that can be chosen depending upon the security needs. The stronger the cipher suite strength, the better the security. However, the strong cipher suite strength can result in performance consequences.

Set the cipher strength groups setting to **Strong**, which is the default, to limit the acceptable ciphers unless you have a good reason not to use this setting. If you want, you can use the pick list to specify precisely the ciphers to use, but selecting specific ciphers is not usually necessary.

Trust and key managers

A *trust manager* is a class that is invoked during SSL handshakes to make trust decisions about remote endpoints requesting connections. The default trust manager, either the IbmX509 or IbmPKIX, is used to validate the signature and expiration of certificates, while additional custom trust managers can be plugged in to perform extended certificate and host name checks.

The IbmX509 trust manager provides basic peer certificate validation based on the trusted signer certificates present in the SSL configuration's truststore. Therefore, we highly recommend that you remove those unverified self-signed signer certificates and default root certificates (that you do not need) from the certificate authorities.

The IbmPKIX trust manager can replace the IbmX509 for trust decisions in an SSL configuration. Standard certificate validation is provided, similar to the IbmX509 trust manager, but it also provides *extended certificate revocation list (CRL) checking*, where it checks that certificates contain CRL distribution points.

You can define a custom trust manager that runs with the default trust manager that you select. The custom trust manager must implement the Java Secure Socket Extension (JSSE) javax.net.ssl.X509TrustManager interface and, optionally, the com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface to obtain product-specific information.

Refer to “Creating a custom trust manager configuration for SSL” at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_sslcreatecustrustmgr.html

A *key manager* is a class that is used during the SSL handshake to retrieve, by alias, the appropriate certificate from the keystore. By default, IbmX509 is the only key manager unless you create a custom key manager.

If you choose to implement your own key manager, you can affect the alias selection behavior, because the key manager is responsible for selecting the certificate alias from the keystore. The custom key manager might not interpret the SSL configuration as the WebSphere Application Server key manager IbmX509 does.

For more information, refer to “Creating a custom key manager for SSL” at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_sslcreatecuskeymgr.html

5. Click **OK**, and then, click **Save** to save the new SSL configuration.

Additional SSL configuration attributes

There are a couple of other configuration options for SSL that are available from the main SSL certificate and key management page.

Federal Information Processing Standard (FIPS)

FIPS support can be enabled by selecting **Use the United States Federal Information Processing Standard (FIPS) algorithms** on the SSL certificate and key management page. When this option is selected, the LTPA implementation uses IBMJCEFIPS. IBMJCEFIPS supports the United States FIPS-approved cryptographic algorithms for Data Encryption Standard (DES), Triple DES, and Advanced Encryption Standard (AES).

Dynamic SSL configuration updates

Dynamic update functionality provides you with greater flexibility and efficiency, because you can change SSL configurations without restarting WebSphere Application Server for the changes to take effect. If you select **Dynamically update the run time when SSL configuration changes occur**, all SSL-related attributes that change are read from the configuration dynamically after they have been saved and then implemented for new connections. For outbound SSL endpoints, all outbound connections inherit the new configuration changes, because new connections are established for each request. For inbound SSL endpoints, only changes that are implemented by the SSL channel are affected by dynamic updates.

5.4 Advanced concepts

This section provides information about advanced concepts and operations in WebSphere.

5.4.1 Changing default chained certificates

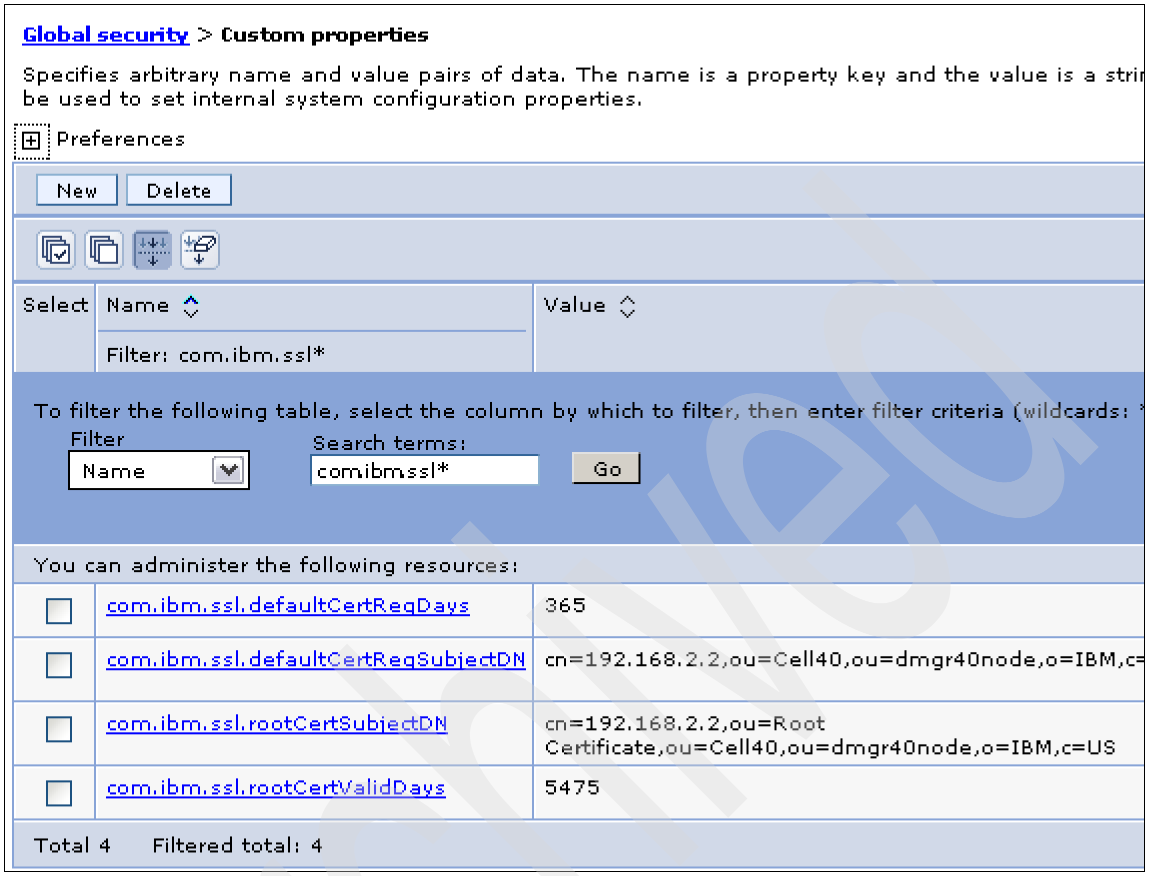
The default chained certificates can be recreated with different information by deleting the `*.p12` files in the appropriate *`profile_home`*`/config` directory structure and from the *`profile_home`*`/etc` directory and then changing the properties that are used to create them to the values that you want the certificates to contain. Then, restart the processes.

The properties of interest are:

- `com.ibm.ssl.defaultCertReqAlias=default_alias`
- `com.ibm.ssl.defaultCertReqSubjectDN=cn=${`*`host name`*`},o=IBM,c=US`
- `com.ibm.ssl.defaultCertReqDays=365`
- `com.ibm.ssl.defaultCertReqKeySize=1024`
- `com.ibm.ssl.rootCertSubjectDN=cn=${`*`host name`*`},ou=Root Certificate, o=IBM,c=US`
- `com.ibm.ssl.rootCertValidDays=7300`
- `com.ibm.ssl.rootCertAlias=root`
- `com.ibm.ssl.rootCertKeySize=1024`

Values that do not exist by default can be added as new custom properties (Figure 5-13 on page 185):

- For a client certificate in `/etc`, you can modify and add the properties in the *`profile_home`*`/properties/ssl.client.props file`.
- For the server certificates in /config, you can modify or add the properties using the administrative console. Click **Security** → **Global security** → **Custom properties**.


Global security > Custom properties

Specifies arbitrary name and value pairs of data. The name is a property key and the value is a stri
be used to set internal system configuration properties.

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *

Filter: Name

Search terms: com.ibm.ssl*

You can administer the following resources:

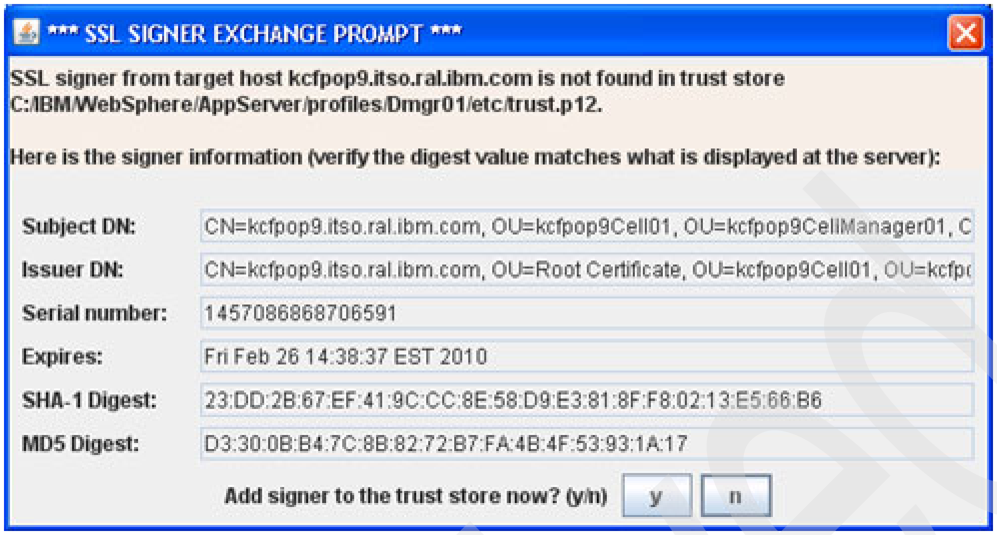
Select	Name (Filter: com.ibm.ssl*)	Value
☐	com.ibm.ssl.defaultCertReqDays	365
☐	com.ibm.ssl.defaultCertReqSubjectDN	cn=192.168.2.2,ou=Cell40,ou=dmgr40node,o=IBM,c=
☐	com.ibm.ssl.rootCertSubjectDN	cn=192.168.2.2,ou=Root Certificate,ou=Cell40,ou=dmgr40node,o=IBM,c=US
☐	com.ibm.ssl.rootCertValidDays	5475

Total 4 Filtered total: 4

Figure 5-13 Global security custom properties for the default chained certificates

After changing the values, you need to restart the cell. If the default certificate does not exist, WebSphere Application Server automatically generates a new default certificate using the new values.

When you restart the deployment manager or node agent, the server's SSL signer has to be added to the client's truststore. By default, the `com.ibm.ssl.enableSignerExchangePrompt` was enabled in ssl.client.props for `"DefaultSSLSettings"`. A GUI will prompt you to accept the signer during the connection attempt (Figure 5-14 on page 186).


*** SSL SIGNER EXCHANGE PROMPT ***

SSL signer from target host kcfpop9.itso.ral.ibm.com is not found in trust store C:/IBM/WebSphere/AppServer/profiles/Dmgr01/etc/trust.p12.

Here is the signer information (verify the digest value matches what is displayed at the server):

Subject DN:	CN=kcfpop9.itso.ral.ibm.com, OU=kcfpop9Cell01, OU=kcfpop9CellManager01, C
Issuer DN:	CN=kcfpop9.itso.ral.ibm.com, OU=Root Certificate, OU=kcfpop9Cell01, OU=kcfp(
Serial number:	1457086868706591
Expires:	Fri Feb 26 14:38:37 EST 2010
SHA-1 Digest:	23:DD:2B:67:EF:41:9C:CC:8E:58:D9:E3:81:8F:F8:02:13:E5:66:B6
MD5 Digest:	D3:30:0B:B4:7C:8B:82:72:B7:FA:4B:4F:53:93:1A:17

Add signer to the trust store now? (y/n)

Figure 5-14 SSL signer exchange prompt

If a default_alias value already exists in the keystore, the run time appends _#, where the number sign (#) is a number that increases until it is unique in the keystore. ${*host name*} is a variable that is resolved to the host name where it was originally created. The default expiration date of chained certificates is one year from their creation date.

5.4.2 Creating and defining a CA client

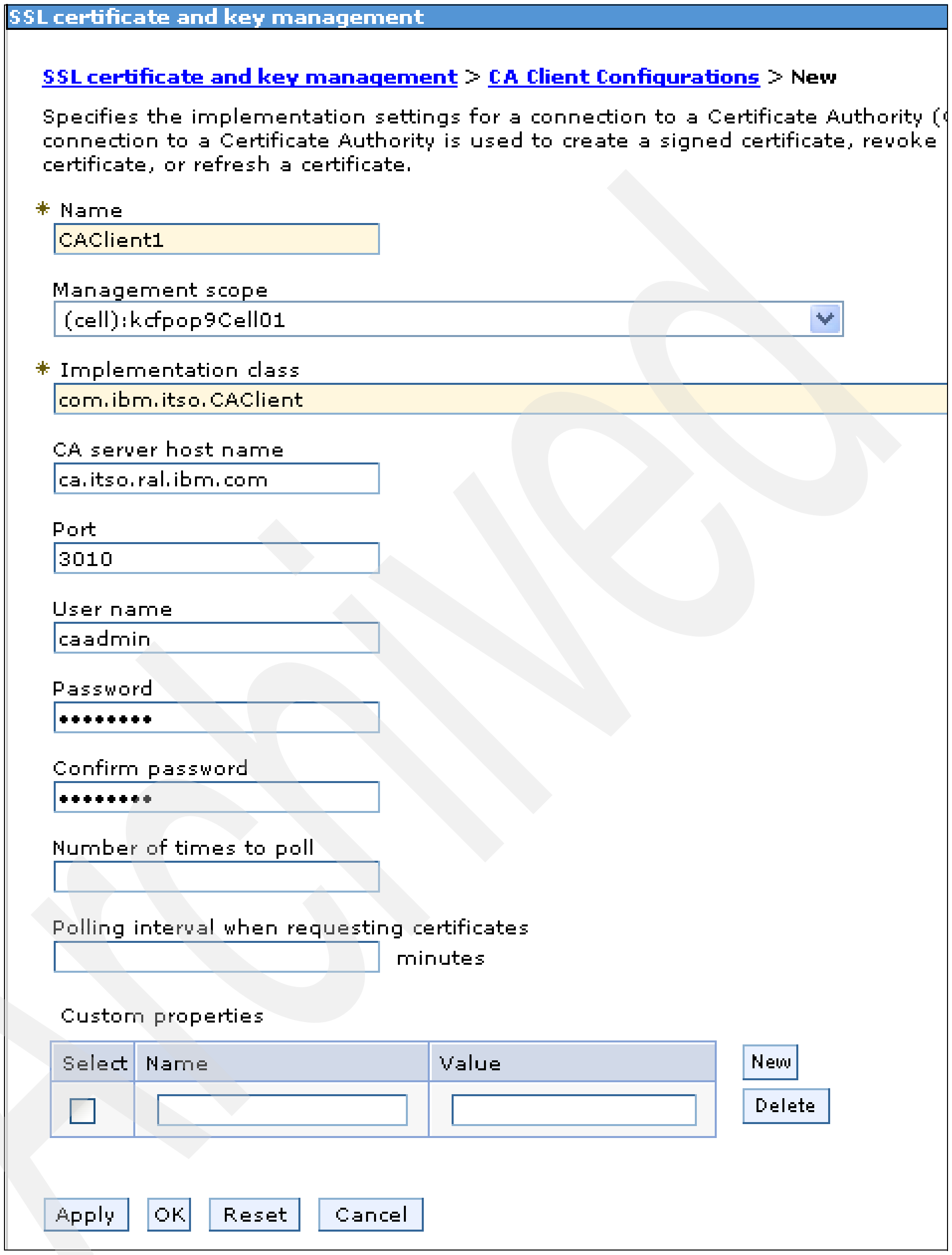
A new interface, WSPKIClient, has been introduced in WebSphere Application Server V7 to allow users to make certificate requests directly to a certificate authority. Information about working with this interface can be found at "Developing a WSPKIClient interface for communicating with a certificate authority" at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_7dev_WSPKIClient_interface.html

A CA client must be created to connect to the CA server before creating a CA certificate. You need to implement the `com.ibm.wsspi.ssl.WSPKIClient` interface to enable WebSphere Application Server security to communicate with a remote CA. The class name needs be provided as part of the CA client when it is created (shown in Figure 5-15 on page 188).

You can create a CA client in the administrative console by using following steps:

1. Click **Security** → **SSL certificate and key management**.
2. Click **Certificate Authority (CA) client configurations**. A panel of existing CA clients appears.
3. Click **New** to create a new CA client in the configuration.


SSL certificate and key management

SSL certificate and key management > CA Client Configurations > New

Specifies the implementation settings for a connection to a Certificate Authority (
connection to a Certificate Authority is used to create a signed certificate, revoke
certificate, or refresh a certificate.

* Name: CAClient1

Management scope: (cell):kcfpop9Cell01

* Implementation class: com.ibm.itso.CAClient

CA server host name: ca.itso.ral.ibm.com

Port: 3010

User name: caadmin

Password: ••••••••

Confirm password: ••••••••

Number of times to poll:

Polling interval when requesting certificates: minutes

Custom properties

Select	Name	Value
☐		

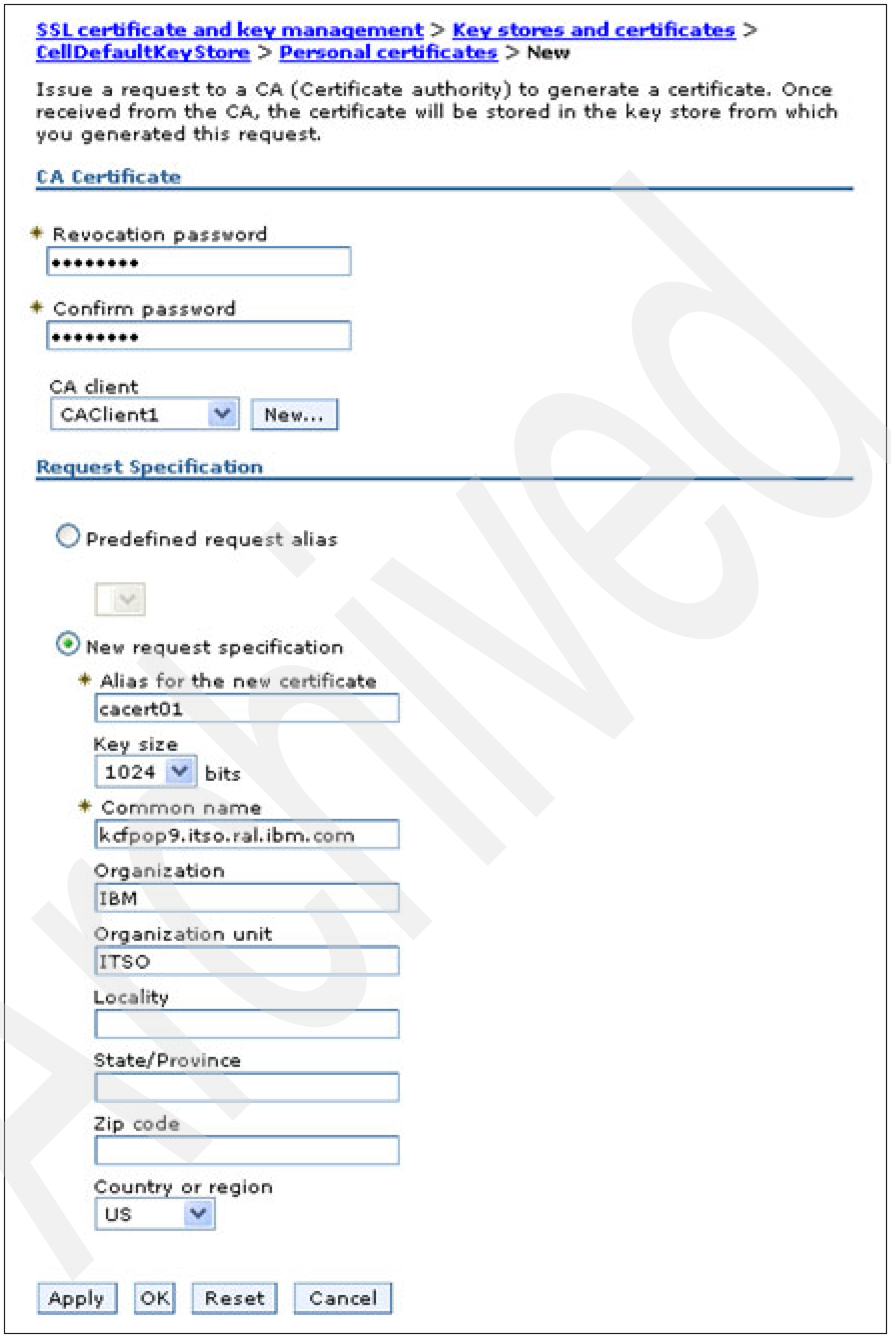
Figure 5-15 New CA client configuration

4. Fill in the information, click **Apply**, and then, click **Save** to save the configuration.

When a CA Client object is configured to connect to the CA, certificates can be created by the CA. These certificates are tracked in the security configuration in an object called CACertificate. The certificate is stored in a keystore, and a CACertificate object is added to the configuration to reference the certificate. CA certificates are personal certificates.

To use the administrative console to create a CA certificate:

1. Click **Security** → **SSL certificate and key management**.
2. Under Related Items, click **keystores and certificates**.
3. Click a ***<keystore name>*** to which you want to add the new CA certificate.
4. Under Additional Properties, click **Personal certificates** to create a new CA certificate in the configuration.
5. Click **Create**, and select **CA-signed Certificate**
6. Fill in the information in the CA certificate section, as shown in Figure 5-16 on page 190.



SSL certificate and key management > Key stores and certificates > CellDefaultKeyStore > Personal certificates > New

Issue a request to a CA (Certificate authority) to generate a certificate. Once received from the CA, the certificate will be stored in the key store from which you generated this request.

CA Certificate

* Revocation password: ••••••••

* Confirm password: ••••••••

CA client: CAClient1

Request Specification

Predefined request alias

[✓ New request specification]

* Alias for the new certificate: cacert01

Key size: 1024 bits

* Common name: kcfpop9.itso.ral.ibm.com

Organization: IBM

Organization unit: ITSO

Locality:

State/Province:

Zip code:

Country or region: US

Figure 5-16   CA Certificate

7. Click **Apply**, and then, click **Save** to save the configuration.

You can also create a CA certificate by using the requestCACertificate AdminTask.

If a CA certificate is compromised and the servers cannot trust it any longer, the CA certificate can be revoked. You can invoke the CA certificate in the manage personal certificates page (Figure 5-16 on page 190).

5.4.3 SSL isolation

SSL enables you to ensure that any client that attempts to connect to a server during the handshake first performs server authentication. Using an SSL configuration at the node, application server, and cluster scopes, you can isolate the communication between servers that are not allowed to communicate with each other over secure ports. The key to isolation is to control the signers that are contained in the truststores that are associated with the SSL configuration. When the client does not contain the server signer, it cannot establish a connection to the server. By default, WebSphere uses chained certificates, and each node has a unique root certificate signer. Because the node shares the same root signer, all of the servers in that node can connect to each other because they share the same root signer.

Authenticating only the server-side of a connection is not adequate protection when you need to isolate a server. Any client can obtain a signer certificate for the server and add it to its truststore. SSL client authentication must also be enabled to enforce the isolation requirements on both sides of a connection.

Isolation requires that you use centrally managed SSL configurations for all or most endpoints in the cell.

For more information about SSL isolation, refer to Secure Sockets Layer node, application server, and cluster isolation in the WebSphere Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_sslisolation.html

5.5 SSL troubleshooting and traces

SSL handshake errors are one of the most common problems that can surface when attempting to set up secure communications in WebSphere. This section provides diagnostic steps that can be performed to identify an incorrect SSL setup, the types of traces to gather to diagnose them, and common problems.

5.5.1 Diagnostic steps

When attempting to diagnose SSL handshake issues, perform the following steps to identify the cause of the problem:

1. Identify the endpoints for the SSL communication. It is important to determine the identity of the SSL client (client initiating the SSL handshake) and the identity of the SSL server (the receiving party in the SSL handshake attempt). Identifying the SSL client and the SSL server is sometimes challenging, because the SSL client can be a J2EE client or a thin client, a Web browser, or a WebSphere process. The SSL server is usually a WebSphere process.
2. SSL handshake issues occur between two endpoints, and the SSL handshake error usually surfaces in the error log of the client. Determine the SSL configuration, keystore, and truststore for the SSL client and the SSL server.
3. Knowledge of the certificates can be used to identify where the setup might be incorrect. Display the certificate information for the signer certificate and the personal certificate (if there is a personal certificate) using the administrative console SSL management, or the ikeyman utility.
4. From the displayed information, check the validity period for the signer certificate on the SSL client side and the SSL server side. Verify that the current date falls within the start date and end date for the certificates. The certificate's start date must precede the current date and must not be expired.
5. From the displayed information, confirm that the SSL client's truststore contains the signer certificate of the server. Verify that the issuer's distinguished name and subject distinguished name for the server's signer certificate on the SSL client side match those of the server personal certificate on the SSL server side.

5.5.2 SSL traces

When SSL errors surface and then when you analyze the certificates and keyrings and the WebSphere setup does not provide enough information to diagnose the problem, you can enable SSL traces in either the WebSphere administrative console or in the Java Virtual Machine (JVM™).

WebSphere SSL trace

When diagnosing WebSphere SSL problems, SSL traces can be enabled in WebSphere and in the JVM.

Follow the path in the administrative console in WebSphere, select **Logging and Tracing** → ***server_name*** → **Change Log Detail Levels**. Click the **Configuration** tab and add the WebSphere trace specification:

```
SSL=all
```

Java JSSE trace

To enable the diagnostic trace for determining the cause of SSL handshake errors, follow the path in the administrative console to **Server types** → **WebSphere Application Servers** → ***server_name*** → **Process Definition** (in the Server Infrastructure section under **Java and Process Management**) → **Java Virtual Machine**.

In the Generic JVM Arguments field, add:

```
-Djavax.net.debug=true -Djava.security.auth.debug=all
```

The Java trace setting cannot be enabled dynamically and requires that you restart the application server to pick up the change.

5.6 Implementation examples

This section contains two examples of implementing SSL in a WebSphere environment.

5.6.1 Securing LDAP communication

When configuring WebSphere Application Server to use an LDAP registry, the default is that WebSphere Application Server connects to the LDAP server using port 389 using LDAP protocol. Information is exchanged in plain text and can be sniffed. There is no way to guarantee that WebSphere Application Server is connected to the correct LDAP server. Securing the connection using SSL can help solve this issue. When SSL is enabled, data exchanged between WebSphere and the LDAP registry is encrypted. Both server and client authentication are supported.

This section provides an example of how to configure SSL for an LDAP connection with server authentication.

The example uses IBM Tivoli Directory Server V6.1 on a Windows platform. The host name is `sys4.itso.ral.ibm.com`. The WebSphere Application Server system is at Fix Pack 7.0.0.1 and runs on a Windows platform with the host name `kcfpop9.itso.ral.ibm.com`.

Creating a self-signed certificate for an LDAP server

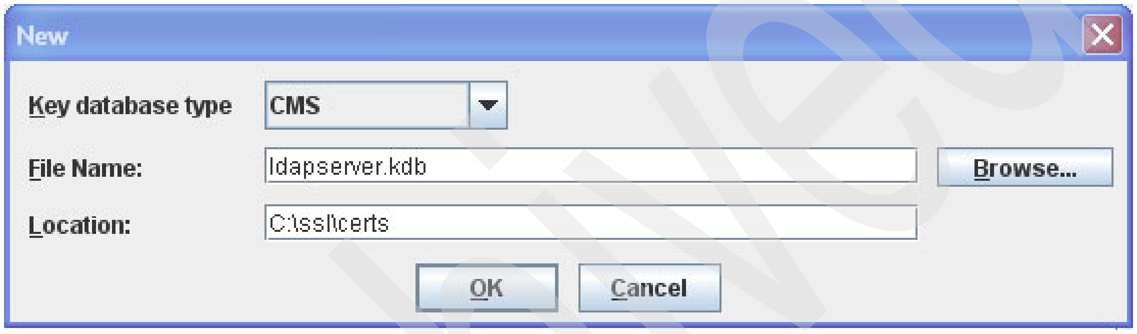
To enable SSL for LDAP, start by creating a self-signed certificate and then extract the certificate to make it available for secure communication.

Complete these steps to create a self-signed digital certificate:

1. Start the iKeyman utility, which is supplied with WebSphere Application Server, IBM HTTP Server, and also, Tivoli Directory Server. For example, in the `C:/ibm/gsk7/bin directory`, type this command:

   ```
   gsk7ikm
   ```

2. On the IBM Key Management page, click **Key Database File** → **New**. Then, Figure 5-17 appears.



Key database type: CMS
File Name: ldapserver.kdb
Location: C:\ssl\certs

Figure 5-17 Create CMS keystore

   On Figure 5-17:

   – Select a default key database type of **CMS**.
   – In the File Name field, type a name for the CMS key database file. For example, type: `ldapserver.kdb`
   – In the Location field, specify a location to store the key database file. For example, type `C:\ssl\certs`

   Click **OK**.

3. On the Password menu that appears, type the password and then confirm the password. Then, select **Stash password to a file**, and click **OK** (Figure 5-18 on page 195).

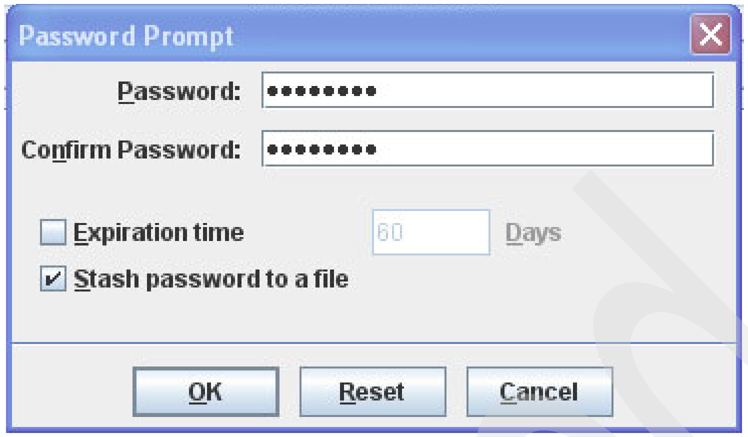
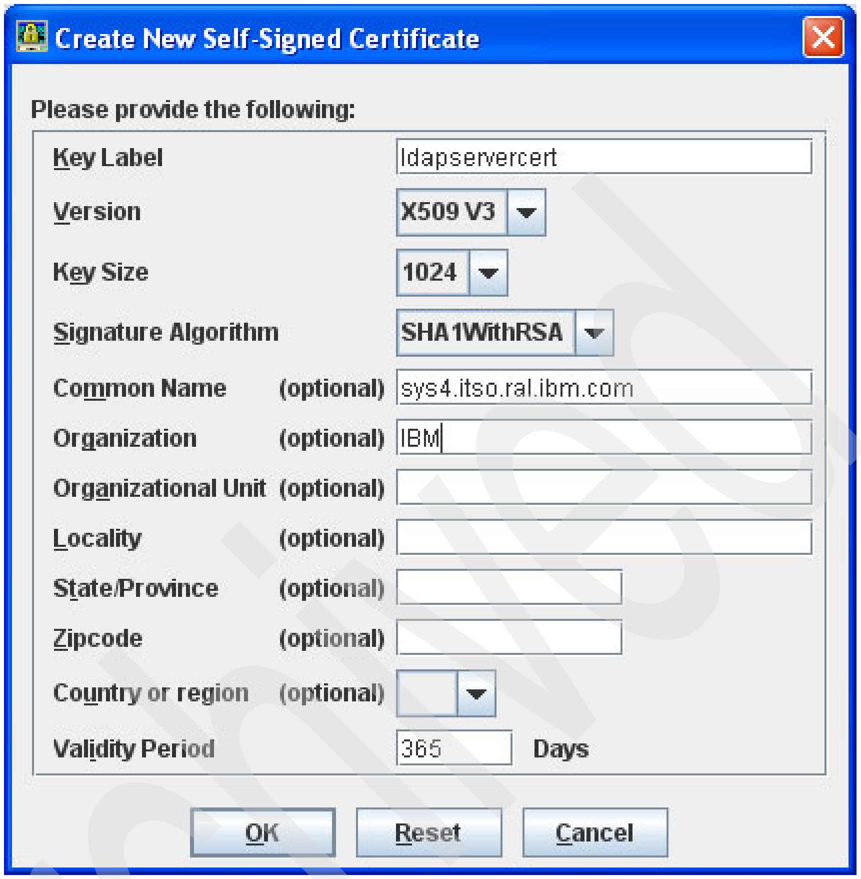


Figure 5-18 Set password for keystore

4. Click **Create** → **New Self Signed Certificate**, and specify a label for the certificate. In this example, enter `ldapservercert`. Refer to Figure 5-19 on page 196.



Please provide the following:

Key Label		ldapservercert
Version		X509 V3
Key Size		1024
Signature Algorithm		SHA1WithRSA
Common Name	(optional)	sys4.itso.ral.ibm.com
Organization	(optional)	IBM
Organizational Unit	(optional)	
Locality	(optional)	
State/Province	(optional)	
Zipcode	(optional)	
Country or region	(optional)	
Validity Period		365 Days

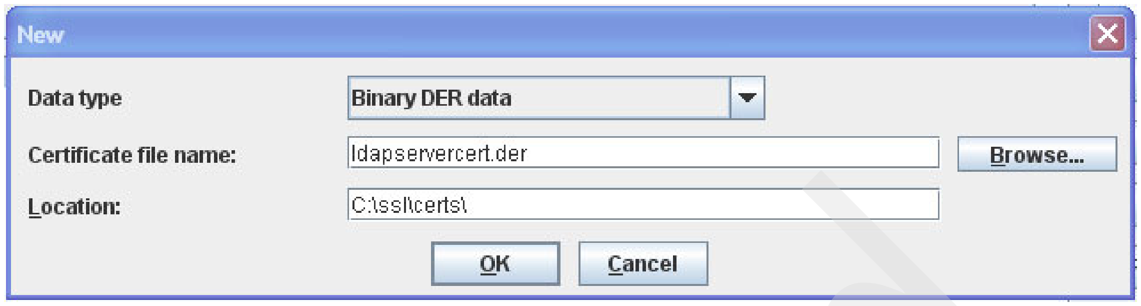
Figure 5-19 Create self-signed certificate

Type the LDAP host name in the Common Name field. You can fill in the remaining fields to match your organization structure. Consider increasing the Validity Period. Click **OK** to generate the personal certificate.

A self-signed certificate now exists, including public and private keys.

5. For subsequent use with WebSphere Application Server, extract the contents of the certificate into a Binary DER Data file.

   Select **Extract Certificate**. Figure 5-20 on page 197 appears.



Data type: Binary DER data

Certificate file name: ldapservercert.der

Location: C:\ssl\certs\

Figure 5-20 Extract the certificate

Follow these steps:

– Specify a data type of **Binary DER Data**. A file with an extension of `.der` contains binary data. Specify this format to extract a self-signed certificate.

– Specify the name of the certificate file name that you created, such as `ldapservercert.der`.

– Specify the location where you previously stored the key database file

Click **OK**.

Verify that the `C:/ssl/certs` directory contains the following files:

- `ldapservercert.der`: The certificate file
- `ldapserver.kdb`: Key database file that has the certificate
- `ldapserver.sth`: Stash file that has the password

Configuring the LDAP server for SSL

The next step in the process is to create the LDAP Directory Interchange™ Format (LDIF) file that is used to configure SSL on the Tivoli Directory Server. For more details, refer to:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc/am61_install319.htm

To configure the LDAP server for SSL:

1. Start the directory server.
2. Create an LDIF file, such as `ssl.ldif`, with the data that is shown in Example 5-1.

Example 5-1 LDIF file data

```
dn:cn=SSL,cn=Configuration
changetype:modify
replace:ibm-slapdSslAuth
```

```
ibm-slapdSslAuth:serverauth
-
replace:ibm-slapdSecurity
ibm-slapdsecurity:SSL

dn:cn=SSL,cn=Configuration
changetype:modify
replace: ibm-slapdSslKeyDatabase
ibm-slapdSslKeyDatabase: C:/ssl/certs/ldapserver.kdb
-
replace:ibm-slapdSslCertificate
ibm-slapdSslCertificate: ldapservercert
-
replace: ibm-slapdSSLKeyDatabasePW
ibm-slapdSSLKeyDatabasePW: key1PassWd
```

Note: The empty lines containing only the - (hyphen) character are required for LDIF file formatting.

3. Place the LDIF file in the `bin` directory of the Tivoli Directory server and present this LDIF file to the directory server with the following command:

   ```
   idsldapmodify -D cn=root -w <passwd> -i ssl.ldif
   ```

4. Restart the server. Ensure that the directory server is listening on secure port 636, as shown in Example 5-2.

Example 5-2 Ensure that the directory server is listening on port 636

```
C:\>netstat -an| grep 636
  TCP    0.0.0.0:636            0.0.0.0:0              LISTENING
```

Configuring SSL on WebSphere Application Server

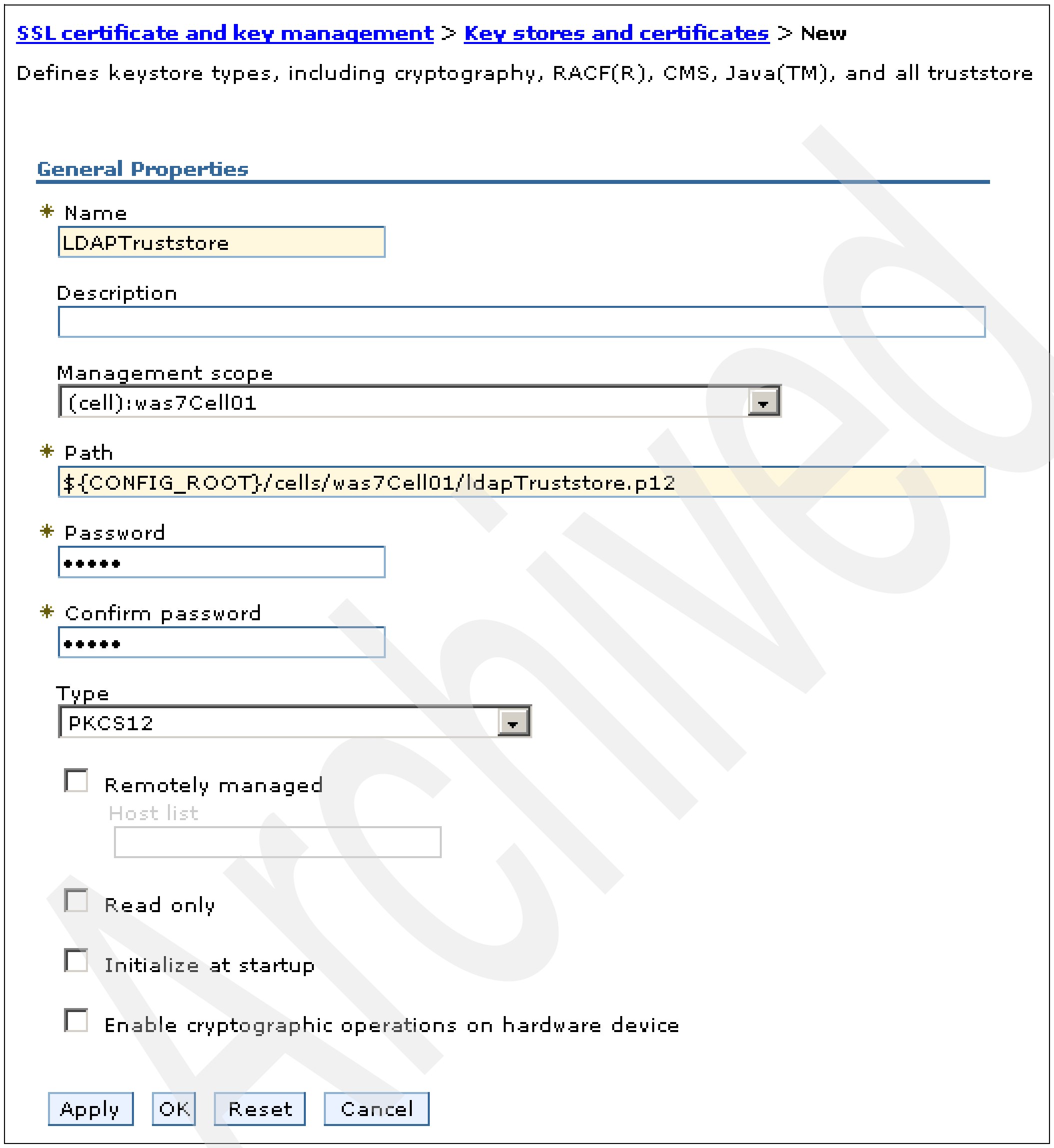
The next step is to configure WebSphere Application Server to enable SSL communication between the LDAP server and WebSphere Application Server:

1. Manually copy the `ldapservercert.der` certificate file from the `C:\ssl\certs` directory on the Tivoli Directory Server to the directory on the WebSphere Application Server.
2. Create a truststore for communication with LDAP. For a Network Deployment environment, we recommend that you place this store at the cell level.

   Perform the following steps to create the truststore:

   a. Click **Security** → **SSL certificate and key management**. In the Related Items section, click **keystores and certificates**.

b. Click **New**, specify the details as shown in Figure 5-21, and click **Apply**.


SSL certificate and key management > Key stores and certificates > New

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore

General Properties

* Name
LDAPTruststore

Description

Management scope
(cell):was7Cell01

* Path
${CONFIG_ROOT}/cells/was7Cell01/ldapTruststore.p12

* Password
•••••

* Confirm password
•••••

Type
PKCS12

☐ Remotely managed
Host list

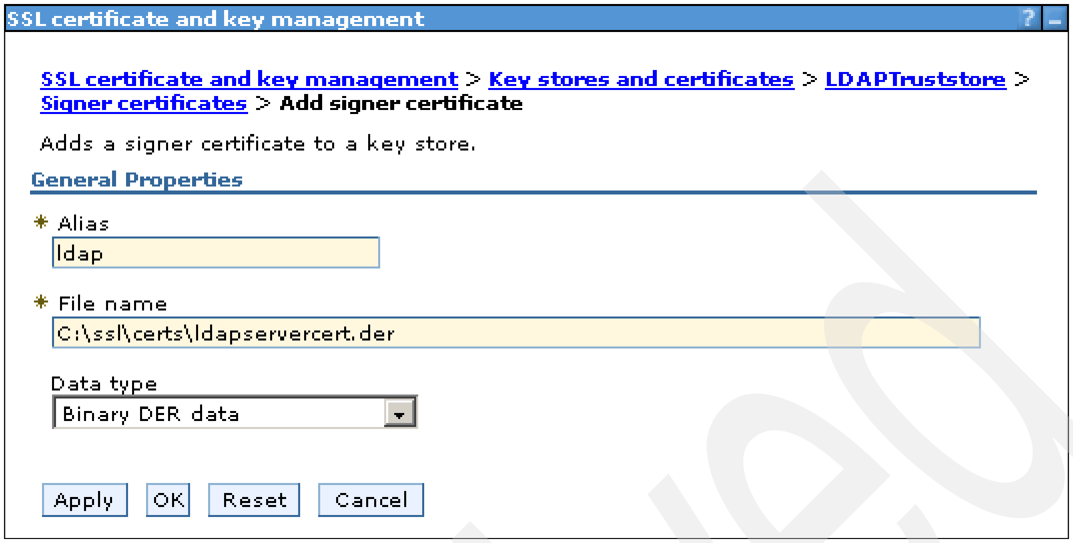
☐ Read only

☐ Initialize at startup

☐ Enable cryptographic operations on hardware device

Figure 5-21 Creating truststore for LDAP connection

3. Add the LDAP's signer certificate to the truststore.
   a. In the Additional Properties section, click **Signer certificates**.
   b. In the signer certificate management page, click **Add** to add the signer certificate, as shown in Figure 5-22 on page 200.


SSL certificate and key management
SSL certificate and key management > Key stores and certificates > LDAPTruststore > Signer certificates > Add signer certificate
Apply
OK
Reset
Cancel

Adds a signer certificate to a key store.

General Properties

* Alias
ldap

* File name
C:\ssl\certs\ldapservercert.der

Data type
Binary DER data

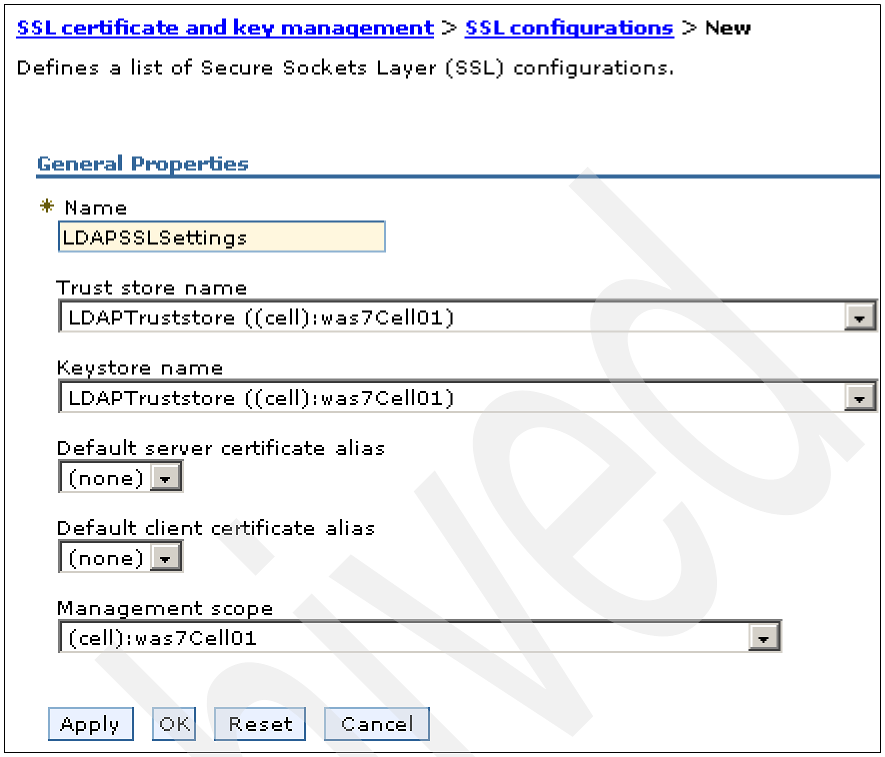
Figure 5-22 Adding signer certificate to truststore

c. Click **OK**, and save the changes.

4. Create a new SSL configuration that will be used for LDAP connection. For the Network Deployment environment, we recommend that you place this configuration at the cell level:

   a. Click **Security** → **SSL certificate and key management**. In the Related Items section, click **SSL Configuration**.

   b. Click **New**, specify the details as shown in Figure 5-23 on page 201, and click **OK**.


SSL certificate and key management > SSL configurations > New

Defines a list of Secure Sockets Layer (SSL) configurations.

General Properties

* Name
LDAPSSLSettings

Trust store name
LDAPTruststore ((cell):was7Cell01)

Keystore name
LDAPTruststore ((cell):was7Cell01)

Default server certificate alias
(none)

Default client certificate alias
(none)

Management scope
(cell):was7Cell01

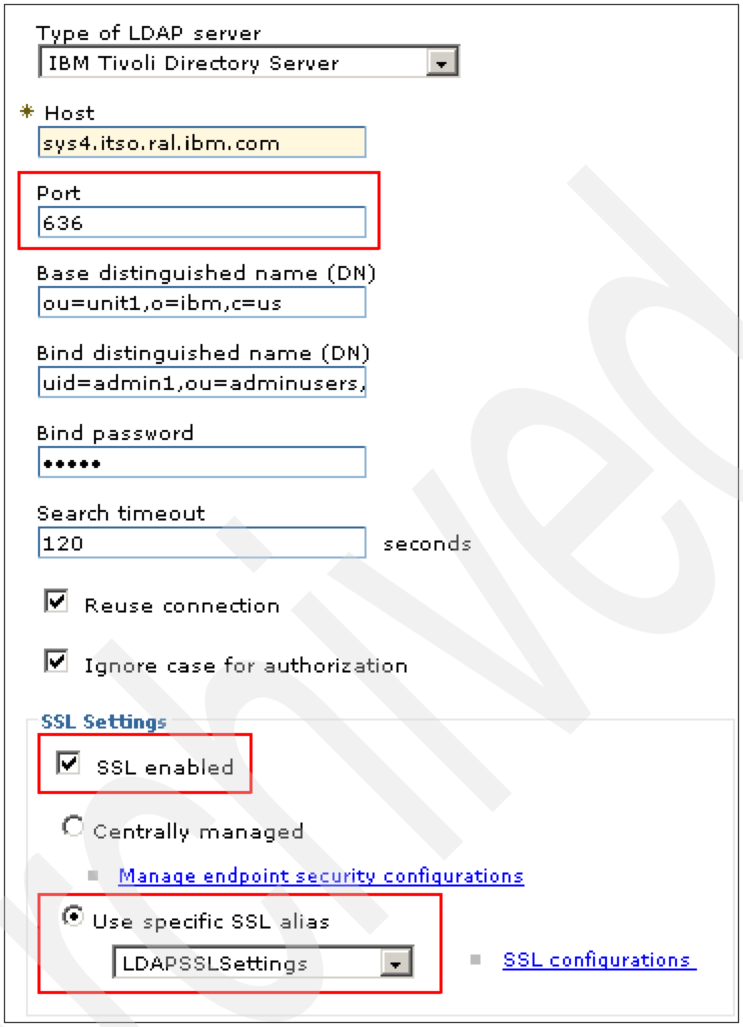
Figure 5-23 Specify the SSL configuration

   c. Save the changes.

5. Enable SSL in the user registry configuration. This section assumes that a stand-alone LDAP registry is used. The configuration for the LDAP repository that is part of the Federated repositories is similar. Follow these steps:

   a. Click **Security** → **Global Security**. In the User account repository, click **Configure**.

   b. Select **SSL enabled**, change the Port to **636** (which is the default SSL secure port), select **Use specific SSL alias**, and choose **LDAPSSLSettings**. These changes are shown in Figure 5-24 on page 202.


Type of LDAP server
IBM Tivoli Directory Server

Host
sys4.itso.ral.ibm.com

Port
636

Base distinguished name (DN)
ou=unit1,o=ibm,c=us

Bind distinguished name (DN)
uid=admin1,ou=adminusers,

Bind password
•••••

Search timeout
120 seconds

[✓] Reuse connection

[✓] Ignore case for authorization

SSL Settings

[✓] SSL enabled

○ Centrally managed

Manage endpoint security configurations

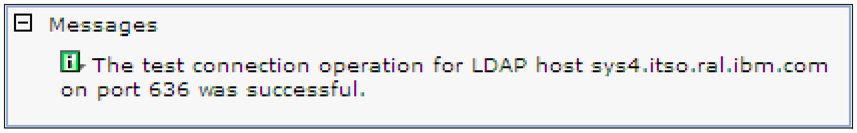
● Use specific SSL alias

LDAPSSLSettings

SSL configurations

Figure 5-24 Enabling SSL for LDAP connection

6. Click **Apply**, and then, click **Save** to save the configuration.
7. Now, you can use **Test Connection** in the top of this page to test the LDAP SSL connection. If the connection is successful, the message will appear as shown in Figure 5-25 on page 203.


Messages

The test connection operation for LDAP host sys4.itso.ral.ibm.com on port 636 was successful.

Figure 5-25 Test connection

Tip: Remember to disable port 389 on the LDAP server to stop non-SSL access.

5.6.2 Securing Web inbound communication

This section illustrates how to configure SSL for connections between the client Web browser and the Web server and how to configure SSL for connections between the Web server plug-in and the application server. It assumes that client certificate authentication is not required.

The WebSphere Application Server HTTP Plug-in is used to establish a connection between a Web server and an application server. When a request is received by the Web server for an application in WebSphere Application Server, the HTTP plug-in will forward it via HTTP. If the request is received via HTTPS, it will be forwarded via HTTPS.

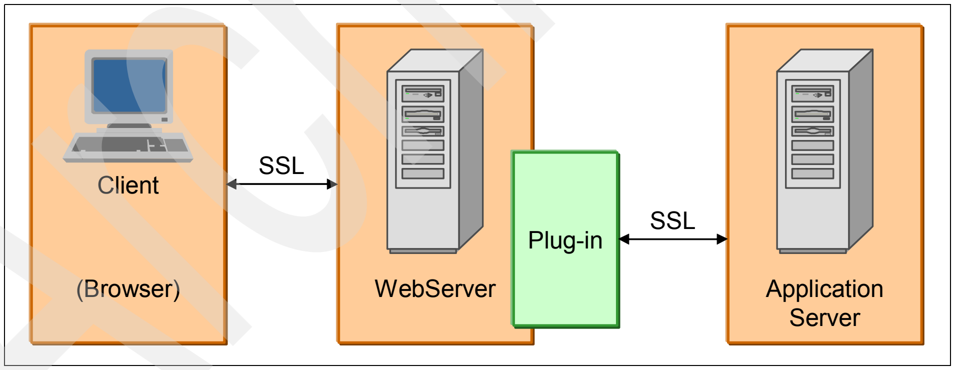


Figure 5-26 SSL connections

Configuring the Web server for SSL

This section illustrates how to implement a secure connection between the browser and the IBM HTTP Web server.

In our example, we use the IBM HTTP Server V7 on a Windows platform.

Follow these steps:

1. Create the key database file and certificates that are needed to authenticate with the Web server during an SSL handshake.

   In this example, we used a self-signed certificate. However, for production installation, we recommend that you use a certificate signed by a well known Certificate Authority. Refer to “Creating a self-signed certificate for an LDAP server” on page 194 for information about using the IBM Key Management Tool to create a self-signed certificate. The IBM Key Management Tool is included with IBM HTTP Server. You can use the following commands to start it:

   ```
   C:\HTTPServer\gsk7\bin>set JAVA_HOME=C:\HTTPServer\java\jre

   C:\HTTPServer\gsk7\bin>gsk7ikm
   ```

   Save this file as `c:\IBM\HTTPServer\ihskey.kdb`

2. Enable the SSL directives in the IBM HTTP Server’s configuration file (`httpd.conf`):

   a. Remove the comment from the the ibm_ssl_module to load this module:

   ```
   LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
   ```

   b. Create an SSL VirtualHost stanza using the following example and directives:

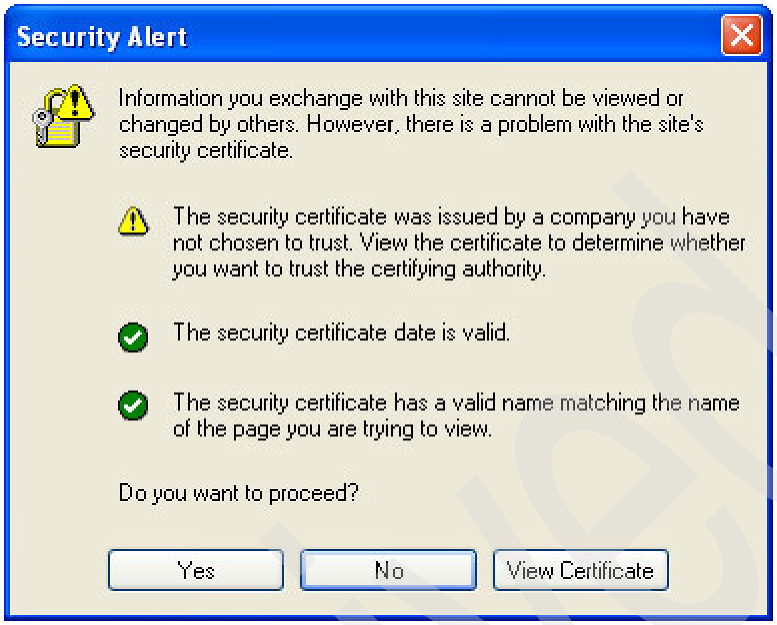
   ```
   Listen 443
   <VirtualHost  *:443>
   SSLEnable
   SSLClientAuth None
   </VirtualHost>
   SSLDisable
   KeyFile "c:/IBM/HTTPServer/ihskey.kdb"
   SSLV2Timeout 100
   SSLV3Timeout 1000
   ```

3. Save the configuration file and restart the IBM HTTP Server.

4. Access the IBM HTTP Server welcome page from a Web browser:

   ```
   https://your-host name
   ```

   You might see a Security Alert page as shown in Figure 5-27 on page 205.


Security Alert

Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

The security certificate date is valid.

The security certificate has a valid name matching the name of the page you are trying to view.

Do you want to proceed?

Figure 5-27 SSL Security Alert

This message occurs, because the configuration uses a self-signed certificate that is not issued by a trust-certifying authority.

5. Click **Yes**. You will see the welcome page.

The next step is to secure the communication between the plug-in and application server with SSL.

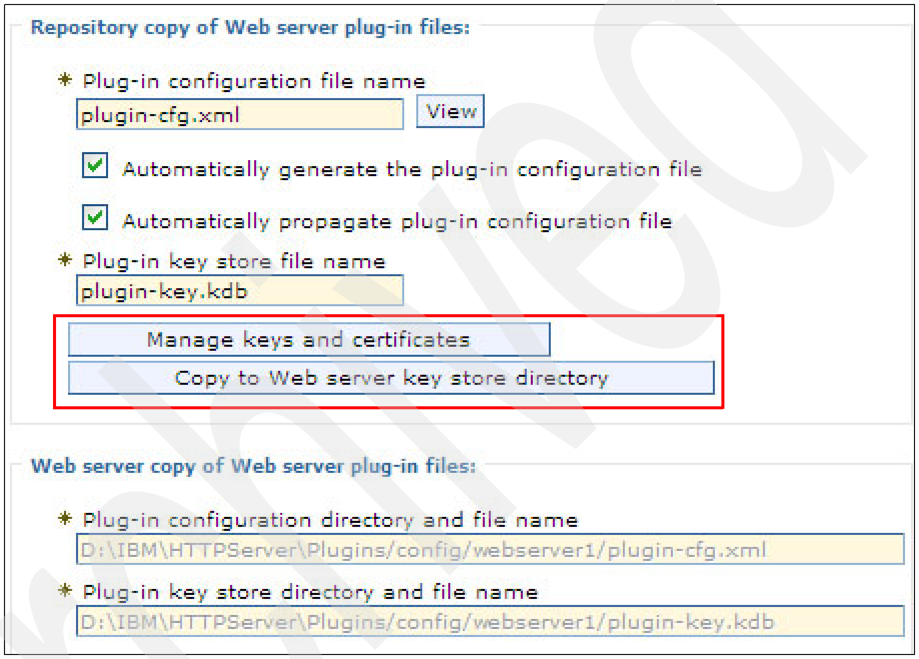
Configuring the plug-in for SSL connection

This section illustrates how to configure the HTTP plug-in to enable SSL connections to the application server.

In this example, we assume that a definition for the Web server has been created in WebSphere Application Server. When the Web server definition was created, WebSphere Application Server associated the Web server plug-in with a Certificate Management Services (CMS) keystore for a specific node. This keystore contains all of the signers for the cell with the self-signed or chained certificate for the node. The plug-in can communicate securely to WebSphere Application Server, even when the plug-in is configured with SSL client authentication enabled, because its personal certificate is signed by the default root certificate, which is trusted by the application server.

Complete this configuration by performing the following steps:

1. From the WebSphere administrative console, click **Servers** → **Server Types** → **Web servers**.
2. Click on the Web server name to open the configuration page.
3. Click **Plug-in properties**.
4. Click **Manage keys and certificates** to see the plug-in keystore details (`CMSKeyStore`) as shown in Figure 5-28.


Repository copy of Web server plug-in files:

* Plug-in configuration file name: plugin-cfg.xml

✓ Automatically generate the plug-in configuration file

✓ Automatically propagate plug-in configuration file

* Plug-in key store file name: plugin-key.kdb

Web server copy of Web server plug-in files:

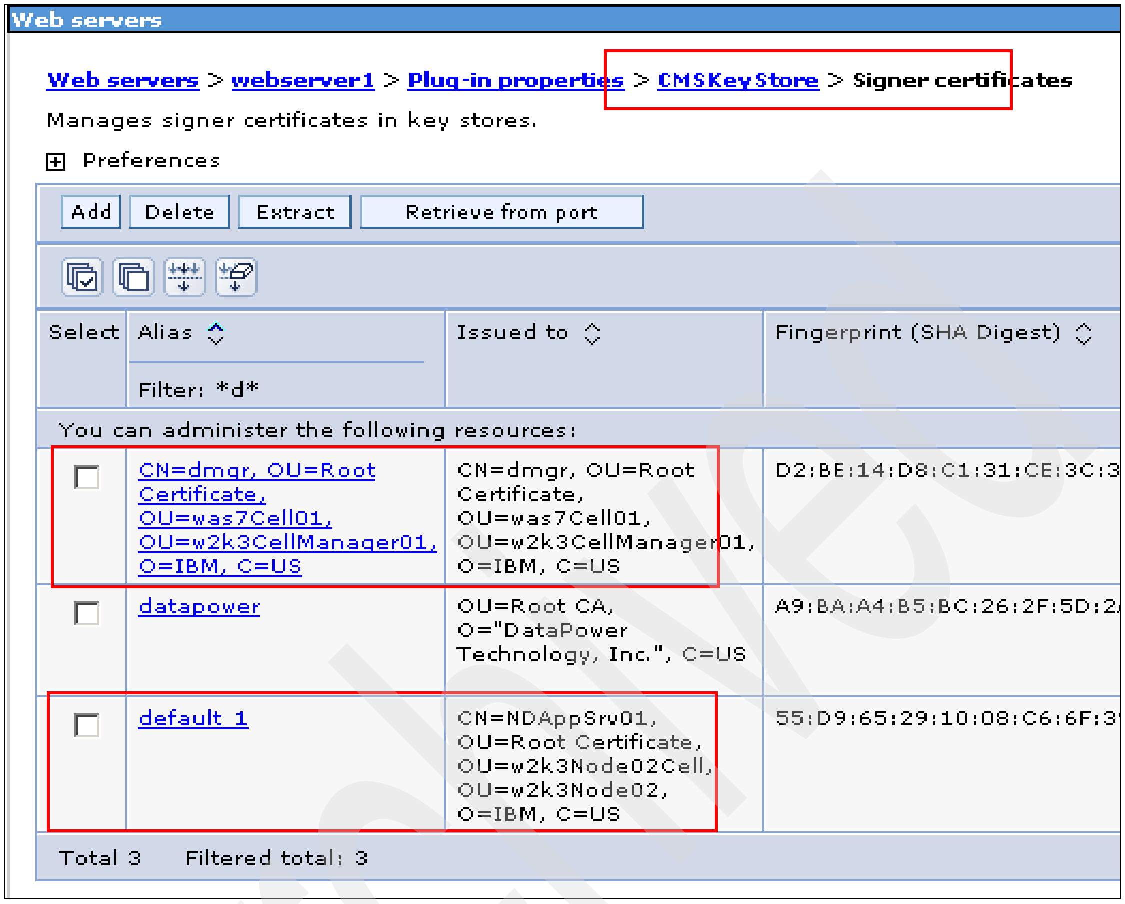
* Plug-in configuration directory and file name: D:\IBM\HTTPServer\Plugins/config/webserver1/plugin-cfg.xml

* Plug-in key store directory and file name: D:\IBM\HTTPServer\Plugins/config/webserver1/plugin-key.kdb

View
Manage keys and certificates
Copy to Web server key store directory

Figure 5-28 Plug-in properties

5. Click **Signer certificates** to see a list of the signer certificates. We have to ensure that all nodes' root certificates are there, as shown in Figure 5-29 on page 207.


Web servers

Web servers > webserver1 > Plug-in properties > CMSKeyStore > Signer certificates

Manages signer certificates in key stores.

Preferences

Select	Alias (Filter: *d*)	Issued to	Fingerprint (SHA Digest)
☐	CN=dmgr, OU=Root Certificate, OU=was7Cell01, OU=w2k3CellManager01, O=IBM, C=US	CN=dmgr, OU=Root Certificate, OU=was7Cell01, OU=w2k3CellManager01, O=IBM, C=US	D2:BE:14:D8:C1:31:CE:3C:3
☐	datapower	OU=Root CA, O="DataPower Technology, Inc.", C=US	A9:BA:A4:B5:BC:26:2F:5D:2
☐	default_1	CN=NDAppSrv01, OU=Root Certificate, OU=w2k3Node02Cell, OU=w2k3Node02, O=IBM, C=US	55:D9:65:29:10:08:C6:6F:3

Total 3 Filtered total: 3

Figure 5-29 Filtered list of signer certificates

The default root certificate and the node root certificate are automatically added to the keystore. If you add other nodes later, their root certificates will be added to this keystore; however, the keystore is not automatically propagated to Web servers.

6. Copy the Web server keystore and stash files to the Web server machine:

   a. If you use managed Web servers, click **Copy to Web server keystore directory** on the plug-in properties page (Figure 5-28 on page 206).

      When you use this option, the plug-in keyring file is propagated from `*node_profile_root*/servers/*web_server_def*/plugin-key.kdb` on the deployment manager system to `*plug-in_root*/config/*web_server_def*/plugin-key.kdb` on the Web server computer.

b. If you use an unmanaged Web server, manually copy the keystore and stash files to the Web server.

c. Ensure that the plug-in configuration points to the location where you stored the files. The following transport directives in the `plug-in.xml` configuration file will contain the location:

```
<Transport Hostname="w2k3" Port="9443" Protocol="https">

<Property Name="keyring"
Value="c:\HTTPServer\Plugins\config\webserver1\plugin-key.kdb"/>

<Property Name="stashfile"
Value="c:\HTTPServer\Plugins\config\webserver1\plugin-key.sth"/>

</Transport>
```

Note that these directives will not be included in the `plug-in.xml` file until you have associated the Web server with an application module.

Note: Ensure that the virtual host that is mapped to the applications includes the SSL port and that the plug-in file is refreshed with the latest configuration.

Summary

At the completion of this process, Web browsers can access applications that are deployed on WebSphere Application Server over connections that are protected by SSL.

6

Common Secure Interoperability Version 2 administration

The Common Secure Interoperability Version 2 (CSIv2) specification is an open standard that defines how to authenticate secure requests for inter- Object Request Broker (ORB) communication.

This chapter discusses CSIv2 security in WebSphere Application Server V7 environments. It contains the following topics:

6.1 Overview of CSIv2

The Object Management Group (OMG) defined the Common Secure Interoperability Version 2 (CSIv2) specification. Detailed documentation about the specification is available here:

`http://www.omg.org/technology/documents/formal/omg_security.htm#CSIv2`

CSIv2 specifies a Security Attribute Service (SAS) that enables interoperable authentication, delegation, and privileges.

CSIv2 SAS as opposed to IBM SAS: Do not confuse the term CSIv2 SAS with the IBM SAS protocol. The IBM SAS protocol is only supported between WebSphere Application Server servers and between Version 6.0.*x* and earlier version servers that have been federated in a Version 7.0 cell. The IBM SAS protocol is deprecated and will be removed in future releases of WebSphere Application Server.

WebSphere Application Server Version 7.0 is compliant with the CSIv2 specification so that it can interoperate with other Java 2 Platform, Enterprise Edition (J2EE) server vendors.

The CSIv2 SAS protocol exchanges its elements in the service context of General Inter-ORB Protocol (GIOP) request and reply messages. It is important to understand this basic request and reply structure when debugging CSIv2 problems. Typically, the client sends a request message to the server, along with the client's CSIv2 configuration. The server receives this request where it has its own CSIv2 configuration. Both the client and server configurations are merged into an effective authentication policy that both sides can comply with. This authentication policy determines the level of security and the type of authentication that occurs between the client and the server.

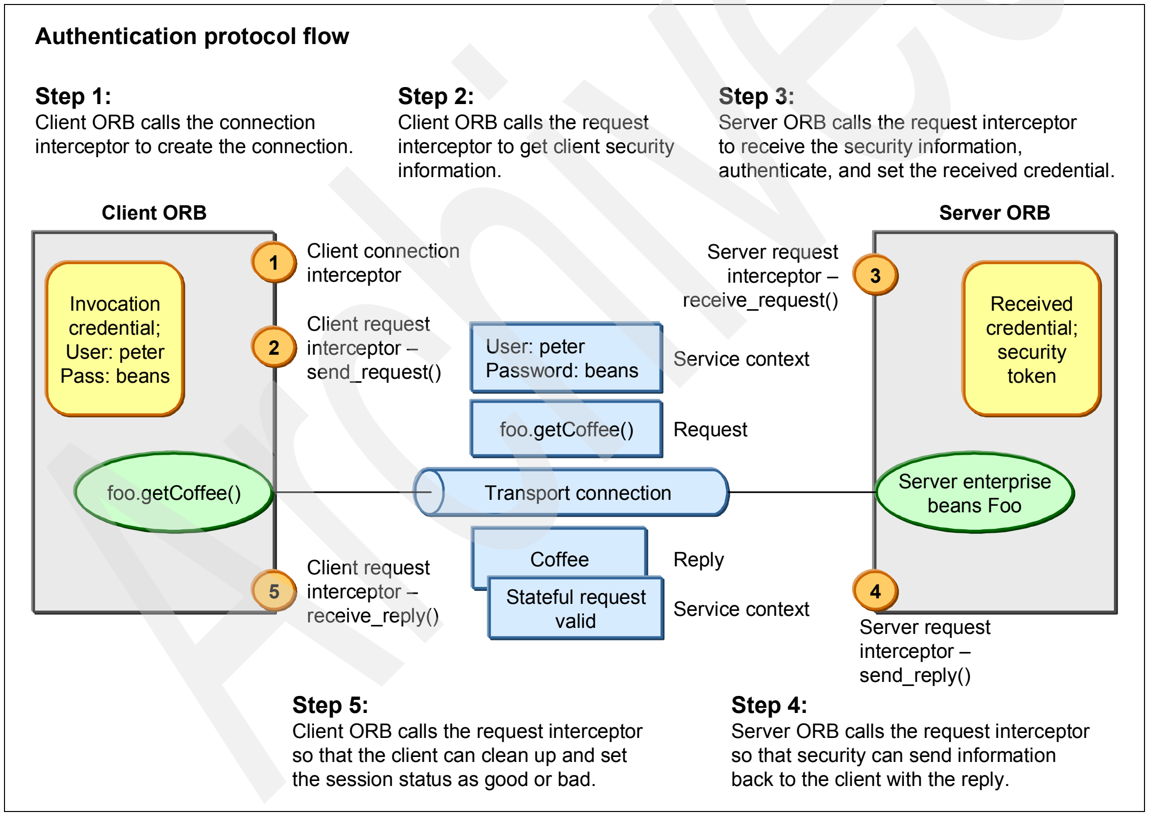
Important: If the authentication protocol cannot come up with a policy that both the client and server can satisfy, the request is not sent.

After the merged authentication policy is established, the server processes the request. The server then sends a reply message back to the client with the outcome of that request. For a more detailed discussion about the flow of this authentication protocol, refer to 6.2, "The CSIv2 authentication protocol" on page 211. For an in-depth look at troubleshooting CSIv2, refer to 6.5, "Troubleshooting CSIv2" on page 240.

6.2 The CSIv2 authentication protocol

The authentication protocol used by WebSphere Application Server is an add-on to the Interoperable Inter-ORB Protocol (IIOP) services. IIOP is a request-and-reply communication protocol that is used to send messages between two Object Request Brokers (ORBs). WebSphere injects into this process by establishing a security context as part of the ORB request. For each request that a client ORB makes to a server ORB, an associated reply is made by the server ORB back to the client ORB. When this communication takes place between two servers, the first server that is sending the request is referred to as the upstream server. The second server that is receiving the request is referred to as the downstream server.

Figure 6-1 illustrates the flow of the authentication protocol.



Authentication protocol flow
Step 1:
Client ORB calls the connection interceptor to create the connection.
Step 2:
Client ORB calls the request interceptor to get client security information.
Step 3:
Server ORB calls the request interceptor to receive the security information, authenticate, and set the received credential.
Client ORB
Invocation credential; User: peter Pass: beans
foo.getCoffee()
1
Client connection interceptor
2
Client request interceptor – send_request()
5
Client request interceptor – receive_reply()
User: peter Password: beans
Service context
foo.getCoffee()
Request
Transport connection
Coffee
Reply
Stateful request valid
Service context
Server ORB
3
Server request interceptor – receive_request()
Received credential; security token
Server enterprise beans Foo
4
Server request interceptor – send_reply()
Step 5:
Client ORB calls the request interceptor so that the client can clean up and set the session status as good or bad.
Step 4:
Server ORB calls the request interceptor so that security can send information back to the client with the reply.

Figure 6-1 The authentication protocol flow for CSIv2

The following steps describe in more detail how the authentication protocol works between a client and a server. The numbers correspond to the numbers in Figure 6-1 on page 211:

1. Client connection interceptor

   The client ORB and the server ORB must first establish a connection over the TCP/IP transport (Secure Sockets Layer (SSL) is a secure version of TCP/IP). The client ORB invokes the authentication protocol's client connection interceptor. This interceptor reads the tagged components in the Interoperable Object Reference (IOR) of the object that is being requested on the server. The client configuration and the server configuration are merged to establish an effective authentication policy. Given this authentication policy, the server ORB returns the strength of the connection. The client ORB makes the appropriate connection, usually over SSL. A connection is thus established between the client and the server.

2. Client request interceptor - send_request()

   The client ORB invokes the authentication protocol's client request interceptor. This interceptor sends security information in addition to what has been established by the transport. This security information can include the following information:

   – A user ID and password token that is authenticated by the server

   – An authentication mechanism-specific token that is validated by the server

   – An identity assertion token. *Identity assertion* is a way for one server to trust another server without the need to reauthenticate or revalidate the originating client. For more information about identity assertion, refer to 6.3.2, "Identity assertion and identity mapping" on page 217.

   The interceptor sends this additional security information with the request message in a GIOP service context. A *service context* has a registered identifier so that the server ORB can identify which protocol is sending the information. After the client request interceptor finishes adding the service context to the message, the message is sent to the server ORB using the send_request method.

3. Server request interceptor - receive_request()

   The server ORB invokes the authentication protocol's server request interceptor to receive the message from the client. The interceptor calls the method receive_request(). This interceptor looks for the service context ID that corresponds to the CSIv2 SAS service context. When a server supports both IBM SAS and CSIv2 SAS, the protocol invokes two different server request interceptors. Each of these interceptors looks for different service context IDs. However, only one interceptor finds a service context for any given request.

If the server's request interceptor finds a service context, it reads the information in the service context. This information contains the client's identity.

If the server's request interceptor does not find a service context, the interceptor looks at the transport connection to see if the client sent a certificate. This process is done when the client and server have client certificate authentication configured. If the interceptor finds a client certificate, it will extract the distinguished name (DN) from the certificate. This name maps to an identity in the user registry. If the certificate does not map to an identity, the server rejects the request. For more information about client certificate authentication, refer to "Transport layer" on page 214.

The server then authenticates the client's identity, whether it is the identity extracted from the service context or the identity from the mapped certificate. If the server determines that the security information from the client is valid, the server creates a credential. If the information is not valid, the server rejects the request by throwing a NO_PERMISSION exception.

However, if the client sends no security information, the server creates an unauthenticated credential. The authorization engine determines if the method requested by the client gets invoked. For an unauthenticated credential to invoke a method on the server, the method must either have no security roles defined or the special Everyone subject mapped to it.

The server request interceptor completes the server authentication either successfully or with an exception.

4. Server request interceptor - send_reply()

   The server request interceptor creates a new reply service context to inform the client request interceptor of the outcome. If the server has configured stateful sessions, a session is created with a stateful context ID that is unique to this connection between the client and the server. For more information about stateful and stateless sessions, refer to 6.3.5, "Stateful and stateless sessions" on page 224. The request interceptor sends a reply by calling the method send_reply().

5. Client request interceptor - receive_reply()

   The client request interceptor receives the reply from the server by calling the method receive_reply(). This service context in the reply message provides information about the outcome of the request. If the request is successful and the client has configured stateful sessions, the client marks the session as valid. The client can then reuse this session for all subsequent requests, without having to resend the authentication information. For more information about stateful and stateless sessions, refer to 6.3.5, "Stateful and stateless sessions" on page 224.

If the outcome is not successful, the client request interceptor examines the error message that is sent by the server. Depending on the Common Object Request Architecture (CORBA) minor codes and messages in the reply service context, the request will simply fail or the client can retry the request. For more information about how errors are handled in CSIv2, refer to 6.3.4, “Error handling” on page 222.

6.3 Features of CSIv2

WebSphere’s compliance with the CSIv2 protocol allows it to take advantage of many features in order to create a flexible yet secure and interoperable environment. These features and the advantages are discussed in detail in the sections that follow.

6.3.1 Three layers for authentication

The CSIv2 protocol provides the client with the flexibility to send authentication information at three layers: the attribute layer, the message layer, and the transport layer. The server chooses only one of these layers and authenticates the information that is provided at this layer. The server makes this decision based on the following order of priority: the attribute layer has the highest priority, followed by the message layer, and then the transport layer. If a client sends authentication information at all three layers, the server will use only the attribute layer. If a client sends information at the message layer and at the transport layer, then the server will use the message layer. The server will only use the transport layer when the attribute layer and the message layer do not exist.

The advantage of having three layers for authentication is that it allows one client to interoperate with multiple servers that might have different security requirements.

Transport layer

The transport layer, which is the lowest layer, can contain an SSL client certificate. This form of authentication is referred to as “client certificate authentication.”

The identity is extracted from the certificate and then mapped to a user in the user registry. The server then creates a credential for that user. The identity of the certificate depends on the server’s current realm. If the realm is “Local operating system,” the identity is the first attribute of the distinguished name (DN) in the certificate.

This attribute is typically the common name. For example, if the client's certificate has the following DN, the identity will be `Lopez`, which maps to the user lopez in the local operating system:

```
cn=Lopez,ou=Accounting,o=AcmeCompany,c=us
```

If the current realm is "Standalone LDAP registry," there are two certificate mapping modes from which to choose that determine how the certificate maps to an entry in the registry. The first mapping mode maps the exact DN to a user in the registry. For the second mapping mode, a certificate filter is defined to match certain attributes in the certificate to certain attributes of a user in the LDAP registry.

Benefit: The transport layer optimizes authentication performance. Because an SSL connection is typically created anyway, there is minimal overhead in sending the client certificate.

The disadvantage of using the transport layer is that the client's keystores must be set up appropriately on every client system.

Message layer

The message layer uses a token to store and receive authentication information on the server. This token can be one of three types:

- Generic Security Service Username Password (GSSUP) token (basic authentication)
- Lightweight Third Party Authentication (LTPA) token
- Kerberos (KRB5) token

The authentication information is sent with the message inside the service context. The server knows which mechanism to use when reading and validating the token by checking the object ID (OID) that is sent by the client. Each authentication mechanism has a corresponding and unique OID.

The GSSUP token is an encoded user ID and password. Supplying this token is also referred to as *basic authentication*. The OID for GGSUP is `2.23.130.1.1.1`. The GSSUP token is considered to be less secure than the mechanism-specific tokens.

There are two mechanism-specific tokens that can be sent to a WebSphere Application Server server. The first is a Lightweight Third Party Authentication (LTPA) token. The LTPA OID is `1.3.18.0.2.30.2`.

Note: When sending an LTPA token to a server that spans another cell, the LTPA keys must be shared between the two cells.

New in V7.0.0.3

Kerberos is the other mechanism-specific token. Although configuring the mechanism requires overhead, the Kerberos token is the most secure and interoperable of the three tokens. The advantages of using Kerberos include:

- ► The Kerberos protocol is an open standard, enabling interoperability with other applications (such as .NET) that support Kerberos.
- ► The Key Distribution Center (KDC) acts as a trusted third party.
- ► Kerberos provides a single sign-on (SSO) end-to-end solution that preserves the original requester's identity. A clear text password is never sent to the downstream server.

The OID for the Kerberos token is `1.2.840.113554.1.2.2`.

A pure Java client can send either a GGSUP token or a Kerberos token. A server acting as a client can send any of the three tokens.

Attribute layer

The attribute layer, which is the highest layer, is used to send additional security information. This information can be custom attributes that are added to the subject or an identity token that is sent from an upstream server. An upstream server uses an identity token to send its already authenticated identity to the downstream server, without having to reauthenticate on the receiving end. This mechanism is known as identity assertion, given that the client is asserting an identity to the server. Because the downstream server is not authenticating the identity token, an additional validation is set in place. The downstream server must trust the upstream server. For more information about identity assertion, refer to 6.3.2, "Identity assertion and identity mapping" on page 217. The other feature available on the attribute layer is the ability to pass the security attribute information of the original authenticated login to the downstream server. Transporting these attributes from one server to another server is called *security attribute propagation*. Refer to 6.3.3, "Security attribute propagation" on page 220 for more information.

Table 6-1 on page 217 summarizes what authentication information can be specified at each layer, and its corresponding order of priority, from highest to lowest.

Table 6-1 CSIv2 Layers

CSIv2 layer	Authentication information
Attribute layer	Uses an identity token to support identity assertion to an upstream server
Message layer	Uses a user ID/password token or an authenticated token with an expiration (LTPA or Kerberos)
Transport layer	Uses the SSL client certificate as the identity

6.3.2 Identity assertion and identity mapping

Identity assertion uses an identity token to identify a client to a server. The benefit of identity assertion is that the downstream server does not have to reauthenticate the asserted identity. For this assertion to occur, however, the downstream server must trust the upstream server.

Tip: The network must have strong protection so that an intruder cannot take advantage of the trust established between the two servers.

The following steps describe in more detail how identity assertion works:

1. The upstream server sets the invocation credential.

   The invocation credential is determined by the RunAs mode of the enterprise bean on the upstream server. The RunAs mode can be set to the originating client identity, the system identity, or a different specified identity.

2. The upstream server sends an identity token along with its server identity to the downstream server.

   The identity token contains the invocation credential. There are five formats for the identity token, and the token is defined by what kind of identity is being asserted. Table 6-2 lists the identity token types.

Table 6-2 Identity token types

Identity token type	Meaning
ITTAbsent	The token is absent, and no identity is asserted.
ITTAnonymous	An unauthenticated identity is asserted.
ITTPrincipalName	A GSS name is asserted.
ITTDistinguishedName	An X5.01 distinguished name is asserted.
ITTX509CertChain	A chain of X.509 certificates is asserted.

In addition to the identity token, the upstream server must send its server identity so that a trust can be established between the two servers. The server identity is sent either in the message layer, which will have the higher priority, or in the transport layer.

If message layer security is supported on both sides, the upstream server sends its server identity along with the identity token. The server identity is sent in a mechanism-specific token, which depends on the settings of the upstream server's realm. When the realm is specified to use an automatically generated server identity, which is the default, then an LTPA token is used. In this case, the server identity is the `uniqueId` part of the `accessId`. When the realm is specified to use a server identity that is stored in the repository, then a GSSUP token is used. In this case, the server identity is the user ID. Note that a GSSUP token is more interoperable than an LTPA token when the downstream server is in another cell.

If message layer security is not supported on both sides, the transport layer must be supported, or else a trust between the two servers cannot be established. In this case, the upstream servers sends a client certificate. The server identity is the distinguished name of the certificate.

z/OS: When the downstream server's realm is the Local Operating System, the server identity is the System Authorization Facility (SAF) ID that is mapped to the certificate. If a certificate is not sent, the server identity is the started task identity of the upstream server's control region.

Table 6-3 summarizes the format of the upstream server identity, depending on what CSIv2 layer is used between the two servers.

Table 6-3 The server identity

CSIv2 layer	**Server identity**
Message layer: LTPA token	uniqueId of the accessId[a]
Message layer: GGSUP token	user ID
Transport layer	Distinguished name of the certificate

a. You do not need to list the server identity in the "Trusted Identities" list for the LTPA token if the LTPA keys are shared between the two servers.

3. The downstream server receives the identity token and the server identity of the upstream server.
4. The downstream server authenticates the server identity of the upstream server.

5. The downstream server determines if it trusts the upstream server.

   Before looking at the identity token, the downstream server first determines if it trusts the upstream server. The downstream server checks if the upstream server's server identity is in the "Trusted identities" list. Refer to Table 6-3 on page 218 to determine the server identity that is sent by the upstream server. Note that you do not need to specify anything in the "Trusted identities" list when the servers are using the automatically generated server identity as long as the LTPA keys are shared between the two servers.

   > **z/OS:** When the downstream server's realm is the Local Operating System, the "Trusted identities" list is not used. Instead, the trust is established by checking if the upstream server's server identity has `UPDATE` access to the `CBIND` class, profile `CB.BIND.`*`<SAFprofilePrefix>`*`.`*`<clusterName>`*.

6. The asserted identity in the identity token is mapped to a user in the realm.

   The downstream server examines the format of the identity token. The mapping of the token to a user in the realm depends on the type of the identity token and the realm of the downstream server. Table 6-4 illustrates this mapping.

Table 6-4 Mapping identity tokens to a user

Realm	Identity token	Mapped user
LDAP	ITTPrincipal	The principal is mapped to the UID field.
	ITTDistinguishedName	The user is determined by the LDAP filters.
	ITTX509CertChain	The user is determined by the LDAP filters.
non-LDAP	ITTPrincipal	The principal maps one-to-one to the user ID field.
	ITTDistinguishedName	The first attribute of the distinguished name is mapped to the user ID.
	ITTX509CertChain	The first attribute of the distinguished name is mapped to the user ID.

> **Note:** The asserted identity must exist in the realm of the downstream server. If it does not exist, additional configuration is required. For more information, refer to "Identity mapping".

7. The mapped credential does not need to be authenticated and can now be used for authorization decisions on the downstream server.

The advantages of using identity assertion are:

- The downstream server does not have to authenticate the asserted identity.
- If there is no common authentication mechanism between the two servers, you can use identity assertion for interoperability.

Identity mapping

Identity mapping is the mapping of a user identity. This mapping is necessary when the downstream server's realm differs from the upstream server's realm, or when mapping within a realm is needed (for example, many to one).

There are two ways that you can map an identity:

- *Outbound identity mapping* maps the current identity to the new identity on the upstream server, before calling the downstream server.
- *Inbound identity mapping* maps the identity on the downstream server.

Deciding which of these two scenarios to use depends upon your environment and requirements. However, it is typically easier to map the user identity before the request is sent outbound for the following reasons:

- You have better knowledge of the source user identity.
- You do not have to share LTPA keys with the other downstream server, because you are not mapping the identity to LTPA credentials. Typically, you are mapping the identity to a user ID and password that are present in the downstream server's realm.

For more information about identity mapping, refer to the following page in the Information center:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_perfidmap.html`

6.3.3 Security attribute propagation

The *security attribute propagation* feature allows WebSphere Application Server to transport security attributes from one server to another server. These attributes can be obtained in the following ways:

- When the server is performing the authentication, it can query the user registry for static attributes, such as the user's groups or email. The subject is then populated with these attributes.
- You can configure a custom login module to populate dynamic security attributes, which might include the user's login time, the identity of the original caller, the location of the original caller, and so on.

- If a reverse proxy server is configured, you can use the corresponding Trust Association Interceptor (TAI) to propagate security attributes across the servers. A reverse proxy server acts as a single entry point of authentication to many disparate servers, including WebSphere Application Server.

WebSphere Application Server provides a token framework to support security attribute propagation. The framework defines plug-in points for various token types, allowing the user to define subject uniqueness, custom serialization and data encryption, and token propagation direction. The following tokens are defined in this interface:

- Single sign-on (SSO) token: The SSO token contains attributes that are sent back to the browser in an HTTP cookie. This token enables the server to perform SSO to other WebSphere Application Servers. Because it is a user-specific token, it is added to the Java Authentication and Authorization Service (JAAS) subject.
- Authentication token: The authentication token serves as the identity of the user and also gets added to the JAAS subject.
- Authorization token: The authorization token carries the privilege attributes that are used by WebSphere Application Server to make authorization decisions.
- Propagation token: A propagation token defines the attributes that must be tracked throughout the invocation. Because the token is not associated with the user, it does not get added to the subject like the other tokens. Instead, it is stored in the thread context.
- Kerberos authentication token (KRBAuthnToken): The Kerberos authentication token contains Kerberos credentials, such as the Kerberos principal name, GSSCredential, and Kerberos delegation credential. This token is created when you authenticate to WebSphere Application Server with either Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) SPNEGO Web or Kerberos authentication.

There are two kinds of propagation available:

- Horizontal propagation: Horizontal propagation uses SSO for Web requests. It propagates security attributes between front-end servers. You can configure a Domain Replication Service (DRS) domain so that the server automatically propagates the security attributes to all the servers in that same domain. You can also use Java Remote Extensions (JMX™) calls to propagate the security attributes, which will work even when the servers are not in the same DRS domain.

- Downstream propagation: Downstream propagation uses the CSIv2 protocol for Java Remote Method Invocation (RMI) over IIOP requests. You can configure not only whether a server can propagate security attributes (outbound), but also whether a server can receive the propagated attributes (inbound).

The advantages of security attribute propagation are:

- You do not need to perform a lookup on the downstream server's registry to get the attributes.
- You can keep track of the original caller information, even when user switches occur because of different RunAs configurations.
- You can have a subject identifier that is more unique than just the user name by using the `uniqueId`, which can be relevant when the dynamic attributes change the context of a user login.

For more information about security attribute propagation, refer to the following page in the Information Center:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_secattributeprop.html`

6.3.4 Error handling

The CSIv2 protocol provides error handling to more easily debug failures or even retry the failing request.

CSIv2 exceptions

Whenever an exception occurs during a CSIv2 request, whether on the client or the server, this exception is converted to a Common Object Request Broker Architecture (CORBA) exception. Instead of sending back a normal reply, an error message is sent back to the requestor indicating the reason for the failure. A CORBA exception has the following fields:

- Message: A more detailed description of the cause of the failure is sometimes included in the exception.
- Vmcid: The Vendor Minor Codeset ID is a hexadecimal 5-digit number that is assigned to a specific vendor. The vmcid for WebSphere Application Server is `0x49424`.
- Minor code: The minor code is a hexadecimal 3-digit number that corresponds to a more specific cause of the exception, including whether the request can be retried.

- Completed: The completed field indicates whether the method was completed before the exception occurred. The values can be `yes`, `no`, or `maybe`. `Maybe` means that the completion status cannot be determined.

Note: The range for security minor codes is 0x49424300-0x494243FF.

All the security-related minor codes are documented in the following Information Center article:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rtrb_securitycomp.html

Example 6-1 illustrates a CORBA exception specific to CSIv2 security.

Example 6-1 A CORBA security exception

```
SystemException class: org.omg.CORBA.NO_PERMISSION, message:
Authentication failed. Could not validate Client Authentication Token
and/or Client Certificates during Identity Assertion;
org.omg.CORBA.NO_PERMISSION: Authentication failed. Could not validate
Client Authentication Token and/or Client Certificates during Identity
Assertion  vmcid: 0x49424000  minor code: 30D  completed:
No
```

In this example, the `vmcid` of `0x49424000` identifies it as an exception specific to WebSphere Application Server. The `minor code` of `30D` indicates that the server identity of the sending server is not on the receiving server's trusted principal list. The `completed: No` field means that the method did not complete before the exception was raised. Authentication is retried.

When an authentication request fails, it is possible in certain situations to automatically retry the request a certain number of times before returning the error back to the client. Examples of when a retry might happen include when you enter an invalid user ID and password, or if the server has an expired credential. The minor code in the exception determines whether a retry is attempted, in conjunction with the CSIv2 configuration property `com.ibm.CORBA.authenticationRetryEnabled` (`true` or `false`). By default, this property is set to `true`, and authentication retries are enabled. Refer to the following article in the WebSphere Information Center to determine which security minor codes are retrievable:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rtrb_securitycomp.html

A retry is by default attempted three times before giving up. This number is determined by the CSIv2 configuration property `com.ibm.CORBA.authenticationRetryCount`.

6.3.5 Stateful and stateless sessions

When a security context is established, you have the option to establish a stateful session or a stateless session. Because a security context can be established on the client side and the server side, each side has its own session and session table. Furthermore, each side can be configured differently and independently.

Stateful sessions

In a stateful session, the security context is established one time and reused for subsequent method requests. The security information only needs to be sent for the first request. After that, a context ID is created that is unique to that connection. This context ID is then used to look up the session in the session table in order to get the stored credential.

The advantage of using stateful sessions is that it can increase performance, because the authentication is performed only one time for the first request. We do not recommend using stateful sessions for environments where authentication is required for every client/server contact.

Stateless sessions

A stateless session exists only for the duration of the request that established the security context. You can choose to use stateless sessions to preserve resources when your environment has so many different connections that the same session rarely is reused.

6.4 Configuring CSIv2

When configuring the CSIv2 settings in WebSphere, it is important to understand the three key concepts that are discussed in the following sections:

- **Client and server configuration**

  A CSIv2 request always involves two parties: the entity sending the request, referred to as the *client*, and the entity receiving the request, referred to as the *server*. A client can be a stand-alone client, for example, a Java client, or it can be a server acting as a client, for example, an upstream server that is sending a request to another downstream server.

You need to be aware of which entities in your environment are the clients and which entities are the servers so that you can correctly configure the CSIv2 settings to allow successful communication over RMI/IIOP.

- **Inbound and outbound configuration**

  Every client or server has two sets of CSIv2 settings: the inbound settings and the outbound settings.

  The inbound CSIv2 settings control how incoming authentication requests are handled. Configure these settings for the server that is receiving authentication requests. The inbound settings are also referred to as the *claims*, because they designate the type of authentication that the server is claiming.

  The outbound CSIv2 settings apply to the requests that are sent outbound. Configure these settings for the client that is performing the authentication requests. Because these settings control how requests are performed, they are also referred to as *performs*.

- **Required, supported, or not supported**

  The CSIv2 protocol lets you choose among three options for authenticating at the different CSIv2 layers: *required*, *supported*, or *not supported*. You specify these options for both the inbound and outbound CSIv2 settings. The definitions for these options differ slightly depending on whether the options are specified for inbound or outbound requests:

  – *Required* means that a certain form of authentication must be performed (when configured for the outbound setting), or must be received (when configured for the inbound settings). If these conditions are not met, the request will fail.

  – *Supported* indicates that a certain form of authentication is performed (for outbound) or accepted (for inbound) when it is present. This option is the most interoperable, because the request will not fail when the certain feature is not present.

  – *Not supported* means that the certain form of authentication will explicitly not be performed (for outbound) or not be accepted (for inbound).

Note: The outbound CSIv2 settings of the client need to be compatible with the inbound CSIv2 settings of the server.

When a client requires a feature, it can talk only to servers that either require or support that feature. When a client supports a feature, it can talk to a server that supports or requires that feature, but it can also talk to servers that do not support the feature.

When a server requires a feature, it can talk only to clients that either require or support that feature. When a server supports a feature, it can talk to a client that supports or requires the feature, but it can also talk to clients that do not support the feature or chose not to support the feature.

For example, for a client to support client certificate authentication, setup is required to either generate a self-signed certificate or to get a self-signed certificate from a certificate authority (CA). Clients might not need to complete these actions; therefore, you can configure this feature as not supported. By making this decision, the client cannot communicate with a secure server that requires client certificate authentication. Instead, this client can choose to use the user ID and password as the method of authenticating itself to the server.

Typically, supporting a feature is the most common way of configuring features. It is also the most successful during run time, because it is more forgiving than requiring a feature. Knowing how secure servers are configured in your environment, you can choose the right combination for the client to ensure successful method invocations and still get the most security. If you know that all of your servers support both client certificate and user ID and password authentication for the client, you might want to require one and not support the other. If both the user ID and password and the client certificate are supported on the client and server, both the user ID and password and the client certificate are performed, but the user ID and password take precedence at the server. This action is based on the CSIv2 specification requirements where the message layer authentication (user ID and password) has higher precedence over the transport layer authentication (client certificates).

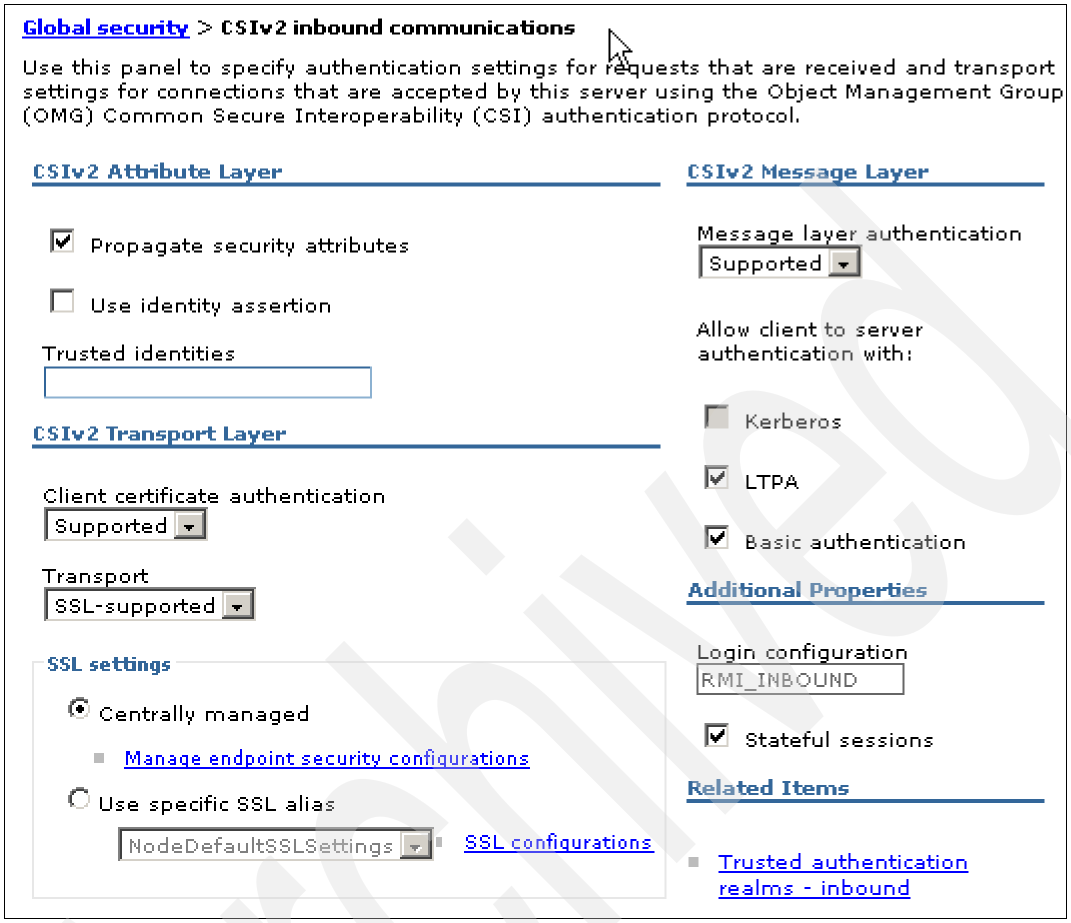
6.4.1 Configuring CSIv2 on a server

To configure CSIv2 settings on a server, you can use the administrative console or wsadmin scripting. We recommend that you initially configure the CSIv2 settings using the administrative console, because the information is more clearly organized and more easily understood.

Administrative console

On the administrative console, the CSIv2 settings are divided into two panels: one panel for the inbound settings and one panel for the outbound settings. These two panels are almost identical, so we discuss them together. Remember that the inbound settings control how the requests that the server receives are authenticated. The outbound settings, however, determine how authentication requests are sent out from the server.

Figure 6-2 on page 227 illustrates the administrative console panel for configuring the CSIv2 inbound communications.



Global security > CSIv2 inbound communications

Use this panel to specify authentication settings for requests that are received and transport settings for connections that are accepted by this server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

CSIv2 Attribute Layer

[✓] Propagate security attributes

[ ] Use identity assertion

Trusted identities:

CSIv2 Transport Layer

Client certificate authentication: Supported

Transport: SSL-supported

SSL settings

(●) Centrally managed

- Manage endpoint security configurations

( ) Use specific SSL alias

NodeDefaultSSLSettings — SSL configurations

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

[ ] Kerberos

[✓] LTPA

[✓] Basic authentication

Additional Properties

Login configuration: RMI_INBOUND

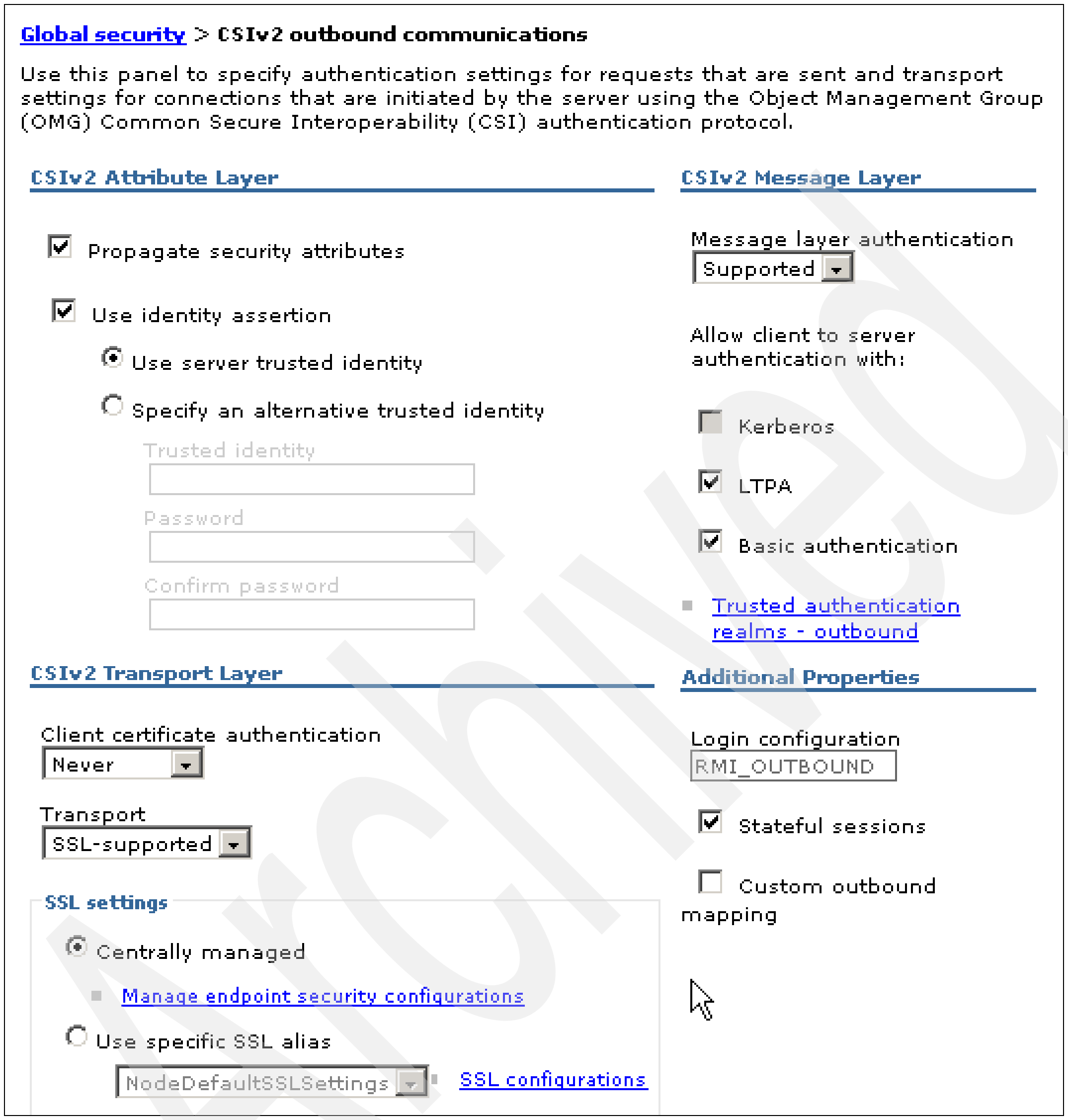
[✓] Stateful sessions

Related Items

- Trusted authentication realms - inbound

Figure 6-2 CSIv2 inbound communication configuration panel

Figure 6-3 on page 228 illustrates the panel for configuring the CSIv2 outbound communications.


Global security > CSIv2 outbound communications

Use this panel to specify authentication settings for requests that are sent and transport settings for connections that are initiated by the server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

CSIv2 Attribute Layer

- [✓] Propagate security attributes
- [✓] Use identity assertion
  - (●) Use server trusted identity
  - ( ) Specify an alternative trusted identity
    - Trusted identity:
    - Password:
    - Confirm password:

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

- [ ] Kerberos
- [✓] LTPA
- [✓] Basic authentication
- Trusted authentication realms - outbound

CSIv2 Transport Layer

Client certificate authentication: Never

Transport: SSL-supported

SSL settings

- (●) Centrally managed
  - Manage endpoint security configurations
- ( ) Use specific SSL alias
  - NodeDefaultSSLSettings   SSL configurations

Additional Properties

Login configuration: RMI_OUTBOUND

- [✓] Stateful sessions
- [ ] Custom outbound mapping

Figure 6-3 CSIv2 outbound communication configuration panel

Notice that in both panels, the security features are visually organized into the three CSIv2 layers. The highest layer of precedence, the CSIv2 attribute layer, is presented first, followed by the message layer, and then, the transport layer.

Attribute layer

In the Attribute Layer section (Figure 6-3 on page 228), you can configure the following settings:

- **Propagate security attributes**: This setting specifies support for security attribute propagation during login requests. When you select this option, the application server retains additional information about the login request, such as the authentication strength used, and retains the identity and location of the request originator. If you do not select this option, the application server does not accept any additional login information to propagate to downstream servers. Refer to 6.3.3, "Security attribute propagation" on page 220 for more information.
- **Use identity assertion:** This setting specifies that identity assertion is used during a downstream Enterprise JavaBeans (EJB) invocation.

  In the outbound configuration, you can additionally specify:

  - **Use server trusted identity:** This setting uses the server identity to establish trust.
  - **Specify an alternative trusted identity:** This setting uses a specified identity to establish trust. You have to provide the username and password.

  In the inbound configuration, you can additionally specify:

  - **Trusted identities**: This setting specifies a pipe-separated (|) list of trusted server administrator user IDs, which are trusted to perform identity assertion to this server, for example, `serverid1|serverid2|serverid3`.

  > **Note:** You can specify the asterisk character (*) on the Trusted Identities list, which implies that all sending servers are trusted. We do not recommend this setting, but it can be useful for debugging.
  >
  > **z/OS:** Specify a semicolon-separated (;) or comma-separated (,) list of trusted server administrator IDs. For example, `serverid1;serverid2` or `serverid1,serverid2`
  >
  > **z/OS:** When the server's realm is the Local Operating System, the Trusted Identities list is not used. Instead, the trust is established by checking that the upstream server's server identity has `UPDATE` access to the `CBIND` class, profile `CB.BIND.<SAFprofilePrefix>.<clusterName>`.

  Identity assertion is performed in the attribute layer and is only applicable on servers. The principal determined at the server is based on precedence rules. If identity assertion is performed, the identity is always derived from the attribute layer. If basic authentication is used without identity assertion, the identity is always derived from the message layer. Finally, if SSL client certificate authentication is performed without either basic authentication or identity assertion, the identity is derived from the transport layer. Refer to 6.3.2, “Identity assertion and identity mapping” on page 217 for more detailed information.

Message layer

In the Message Layer section (Figure 6-3 on page 228), you can configure following settings:

- **Message layer authentication:** This setting defines requirements:
  - **Never:** This setting specifies that this server cannot accept any authentication mechanisms.
  - **Supported:** This setting specifies that client/server can specify the selected authentication mechanisms. However, a method might be invoked without this type of authentication if the other side does not support the selected mechanism. For example, an anonymous or client certificate might be used instead.
  - **Required:** This setting specifies that clients must authenticate using one of the specified mechanisms.
- *(New for V7)* **Allow client to server authentication with:** This option defines available authentication mechanisms:
  - **Kerberos**: This option enables authentication using the Kerberos token. The Kerberos authentication mechanism must be configured first for this option to be available.
  - **LTPA**: This option enables authentication using the LTPA token.
  - **Basic authentication:** This type of authentication typically involves sending a user ID and a password from the client to the server for authentication. This option is also know as *Generic Security Services Username Password (GSSUP)*.

  For more information about authenticating at the message layer, refer to “Message layer” on page 215.

Transport layer

In the Transport Layer section (Figure 6-3 on page 228), you can configure the following settings:

- **Client certificate authentication:** This setting defines if SSL certificates are used for authentication:
  - **Never**: This option specifies that clients cannot attempt client certificate authentication with this server (inbound). The server does not attempt client certificate authentication with downstream servers (outbound).
  - **Supported**: This option specifies that clients can authenticate using SSL client certificates. However, the server can invoke a method without this type of authentication. For example, anonymous or basic authentication can be used instead.
  - **Required**: This option specifies that clients must authenticate using SSL client certificates before invoking the bean method.
- **Transport**: This setting defines the available transports:
  - **TCP/IP**: Client/server supports only TCP/IP; SSL connections are not available.
  - **SSL supported**: Client/server supports SSL and TCP/IP connections. Transport is negotiated during connection.
  - **SSL Required**: The connection must be established over SSL.

  Depending on the Transport setting, CSIv2 can have as few as one listener port and as many as three listener ports. You can open one port for just TCP/IP or when SSL is required. You can open two ports when SSL is supported, and open three ports when the SSL and the SSL client certificate authentication is supported.

  Port names that are used by ORB are:

  - `CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS`: CSIv2 client authentication SSL port
  - `CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS`: CSIv2 SSL port
  - `ORB_LISTENER_ADDRESS`: TCP/IP port

  z/OS: Only one port is opened for SSL: `ORB_SSL_LISTENER_ADDRESS`. The port names `CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS` and `CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS` are not used.

- **SSL Settings:** These settings define the SSL configuration that will be used during SSL connections:

– **Centrally managed**: This option uses the centrally managed SSL configuration.
– **Use specific SSL alias**: This option allows you to select a specific SSL configuration from the list.

> **Note:** We highly recommend that you configure both the inbound and outbound CSIV2 transports with *SSL required* in a secured environment. By default, WebSphere negotiates a mutually acceptable level of transport security. However, if a client requests a non-SSL connection, unless *SSL required* is configured, a non-secure connection is established.

The following settings are in the Additional Properties and Related Items sections:

- Login configuration: This read-only property shows the system login configuration that is used for authentication.
- Stateful sessions: Selecting this option allows you to reuse the security information on all subsequent contacts with valid sessions. Refer to "Stateful and stateless sessions" on page 224 for more information.
- Custom outbound mapping: Select this setting to enable a call to the RMI_OUTBOUND login configuration. If the Security Attribute Propagation option is selected, WebSphere Application Server is already using this login configuration and you do not need to enable custom outbound mapping.

  You might need to modify the RMI_OUTBOUND login configuration if you want to perform outbound identity mapping.
- Trusted authentication realms: If you want to use RMI/IIOP communication across different realms, you have to add external realms as trusted. If you use Kerberos authentication and have a Kerberos cross-realm trusted setup, you also have to add external Kerberos realms as trusted.

New for V7 wsadmin scripting function

You can use AdminTasks in the wsadmin scripting framework to easily configure and view the CSIv2 settings. The main AdminTask groups that you can use are:

- *configureCSIInbound*: This task configures the CSIv2 inbound authentication for a security domain or for the global security configuration.
- *configureCSIOutbound*: This task configures the CSIv2 outbound authentication for a security domain or for the global security configuration.
- *getCSIInboundInfo*: This task displays the current settings for CSIv2 inbound communications in a security domain or in the global security configuration.

- *getCSIOutboundInfo*: This task displays the current settings for CSIv2 outbound communications in a security domain or in the global security configuration.
- *unconfigureCSIInbound*: This task only applies to a security domain. It removes the CSIv2 inbound settings that are defined at the security domain level. This way, the inbound settings of the global security configuration are used instead.
- *unconfigureCSIOutbound*: This task only applies to a security domain. It removes the CSIv2 outbound settings that are defined at the security domain level. This way, the outbound settings of the global security configuration are used instead.

Tip: Use the interactive mode the first time that you use an AdminTask.

The interactive mode prompts you for all the possible parameters of the task, letting you know which ones are required and also displaying suggested values for the parameters. When you complete the task in interactive mode, the entire **`wsadmin`** command is generated and printed out, so you can use it in the future without the interactive mode. This approach helps you avoid scripting and syntax errors.

For a complete description of the CSIv2-related AdminTasks, refer to the following article in the Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rxml_7securityconfig.html

6.4.2 Configuring CSIv2 on a stand-alone client

When you use a secure stand-alone client, such as a Java client, you need to specify a properties file that contains a list of CSIv2 settings. These settings determine how the client will authenticate to a server. Typically, this file is specified with the following JVM property:

```
-Dcom.ibm.CORBA.ConfigURL
```

A sample properties file, `sas.client.props`, is provided and is located in the *`profile_root`*`/properties` directory.

Tip: When you use thin or thick clients, you do not need to specify the `ConfigURL` property. These clients will automatically use this file: *`profile_root`*`/properties/sas.client.props`

Example 6-2 on page 234 shows how to specify the properties file.

Example 6-2 Specifying the ConfigURL property

```
-Dcom.ibm.CORBA.ConfigURL=/WebSphere/AppServer/profiles/default/propert
ies/sas.client.props
```

The `sas.client.props` file consists of several properties, which we explain next, along with their default values (marked with an asterisk character (*):

- Client Security Enablement:
  - `com.ibm.CORBA.securityEnabled (true*, false)`

    This property determines if client security is enabled. If the server's global security is enabled, this property must be set to `true`. Otherwise, the client cannot access any of the secured remote EJB resources.
- Authentication Configuration:
  - `com.ibm.CORBA.authenticationTarget (BasicAuth*, KRB5)`

    For `BasicAuth`, the username and password are sent to the server for message layer authentication.

    For `KRB5`, a Kerberos token is sent to the server, depending on what the `com.ibm.CORBA.loginSource` is set to.
  - `com.ibm.CORBA.authenticationRetryEnabled (true*, false)`

    This property determines whether a failed authentication is retried. Only those failures that are known to be correctable are retried. This option is valid when `com.ibm.CORBA.validateBasicAuth` is set to `true`.
  - `com.ibm.CORBA.authenticationRetryCount (an integer value, 3*)`

    This property determines how many retries are to be attempted for a failed login when `com.ibm.CORBA.authenticationRetryEnabled` is set to `true`.
  - `com.ibm.CORBA.validateBasicAuth (true*, false)`

    This property determines if the user details are authenticated immediately or deferred until the first method request is communicated to the server, when the `com.ibm.CORBA.authenticationTarget` property is set to `BasicAuth`.
  - `com.ibm.CORBA.securityServerHost`

    This property is the name (or IP address) of the security server to validate the user ID and password.
  - `com.ibm.CORBA.securityServerPort`

    This property is the port number of the security server.

- com.ibm.CORBA.loginTimeoutd(an integer value, 300*)

  This property is an integer within a range of 0 to 600. It is the amount of time, in seconds, that the login prompt is available before the login is considered invalid.

- com.ibm.CORBA.loginSource (prompt*, stdin, none, properties, krb5Ccache, krb5Ccache:prompt, krb5Ccache:properties, krb5Ccache:stdin)

  This property determines how the authentication requests interceptor logs if it does not find an invocation credential set:

  - prompt displays a window requesting a user name and password.
  - stdin displays a command line prompt requesting user details.
  - none must be selected if the client uses programmatic login.

    z/OS: When the com.ibm.CORBA.loginSource is set to none, the credentials of the logged-in user are used.

  - properties retrieves the user details from the com.ibm.CORBA.loginUserid and com.ibm.CORBA.loginPassword properties.

  When the com.ibm.CORBA.authenticationTarget is KRB5, these additional options are available:

  - krb5Ccache uses the Kerberos credential cache file.
  - krb5Ccache:prompt uses the Kerberos credential cache file first. If it fails, it falls back to prompt.
  - krb5Ccache:properties uses the Kerberos credential cache file first. If it fails, it falls back to properties.
  - krb5Ccache:stdin uses the Kerberos credential cache file first. If it fails, it falls back to stdin.

- com.ibm.CORBA.loginUserid

  The user ID that is used when the com.ibm.CORBA.loginSource property is set to properties.

- com.ibm.CORBA.loginPassword

  The user password that is used when the com.ibm.CORBA.loginSource property is set to properties.

- `com.ibm.CORBA.krb5ConfigFile`

  This property specifies the location of the Kerberos configuration file as a URL. If this property is not set, the default Kerberos configuration file is used, whose location depends on the operating system of the server:

  - Windows: `C:\winnt\krb5.ini`
  - Linux: `/etc/krb5.conf`
  - UNIX, including z/OS: `/etc/krb5/krb5.conf`
  - IBM i: `/QIBM/UserData/OS400/NetworkAuthentication/krb5.conf`

  This property is only valid when the `com.ibm.CORBA.authenticationTarget` is `KRB5`.

- `com.ibm.CORBA.krb5CcacheFile`

  This property specifies the location of the Kerberos credential cache file as a URL. This property only applies when the `com.ibm.CORBA.loginSource` property is set to `krb5Ccache` and `com.ibm.CORBA.authenticationTarget` is `KRB5`. If this property is not set, the default Kerberos credential cache file is used, whose location is determined by checking the following items in the following order:

  i. The file referenced by the JVM property `KRB5CCNAME`
  ii. *<user.home>*/krb5cc_*<user.name>*
  iii. *<user.home>*/krb5cc (if *user.name* cannot be resolved)

- `com.ibm.CORBA.loginRealm`

  This property specifies the realm of the server with which the client is communicating. Multiple realms can be listed by using the pipe "|" separator. This property only applies when `om.ibm.CORBA.loginSource` property is set to `properties`. You can use this property, for example, when you have multiple security domains with different realms defined on the server.

- CSIV2 add-on authentication protocol

  Certain security properties have supported or required property pairs. The required properties take precedence over the supported properties pair. Therefore, if the required property is enabled, communication with the server must satisfy this property:

Note: A stand-alone client only specifies CSIv2 settings for *outbound* requests; thus, all the CSIv2 properties begin with `com.ibm.CSI.perform`.

- `com.ibm.CSI.performStateful (true*, false)`

  It determines whether the client supports the stateful or stateless session.

- `com.ibm.CSI.performClientAuthenticationRequired (true*, false)`
- `com.ibm.CSI.performClientAuthenticationSupported (true*, false)`

  When *supported*, message layer client authentication is performed when communicating with any server that supports or requires authentication. Message layer client authentication transmits a user ID and password if the `authenticationTarget` property is `BasicAuth`, or it transmits a credential token if the `authenticationTarget` property is one of the token-based mechanisms, for example, Lightweight Third Party Authentication (LTPA) or Kerberos.

  When *required*, message layer client authentication must occur when communicating with any server. If the transport layer authentication property is also enabled, both authentications are performed. However, the message layer client authentication takes precedence at the server side.

- `com.ibm.CSI.performTLClientAuthenticationRequired (true*, false)`
- `com.ibm.CSI.performTLClientAuthenticationSupported (true*, false)`

  When *supported*, transport layer client authentication can be performed, and the client sends the digital certificate to the server during the authentication stage.

  When *required*, the client only authenticates with servers that support transport-layer client authentication.

- `com.ibm.CSI.performTransportAssocSSLTLSRequired (true*, false)`
- `com.ibm.CSI.performTransportAssocSSLTLSSupported (true*, false)`

  When *supported*, the client can use either TCP/IP or SSL to communicate with the server.

  When *required*, the client only communicates with servers that support SSL.

- `com.ibm.CSI.performMessageIntegrityRequired (true*, false)`
- `com.ibm.CSI.performMessageIntegritySupported (true*, false)`

  These properties are only valid when SSL is enabled.

  When *supported*, it can make an SSL connection either with 40-bit ciphers or with digital-signing ciphers.

  When *required*, the connection fails if the server does not support 40-bit ciphers.

  - `com.ibm.CSI.performMessageConfidentialityRequired (true*, false)`
  - `com.ibm.CSI.performMessageConfidentialitySupported (true*, false)`

    These properties are only valid when SSL is enabled.

    When *supported*, it can make SSL connection either with 128-bit ciphers or with a lower encryption strength.

    When *required*, the connection fails if the server does not support 128-bit ciphers.

  - `com.ibm.ssl.alias=DefaultSSLSettings`

    This property specifies the SSL configuration alias that is referenced in the `ssl.client.props` file.

- Additional CORBA configuration:
  - `com.ibm.CORBA.requestTimeout (integer value, 180*)`

    This property specifies the timeout period, in seconds, for responding to requests that are sent from the client. Be careful when specifying this property, and set it only if the application is experiencing problems with timeouts.

6.4.3 CSIv2 considerations in special scenarios

There are certain scenarios where you have to be aware of the implications of the various CSIv2 settings. The two most common scenarios are:

- **cross-cell:** The client and server are part of two different cells.
- **cross-registry:** The client and server authenticate to two different registries.
- **cross-system:** The client and server are on two different machines (only relevant on z/OS).

Cross-cell communication using LTPA

When your client and server are part of two different cells, you need to be aware that each cell has its own set of LTPA keys and its own set of keystores and truststores.

If the client is sending an LTPA token to the server, the server must be able to decrypt the token in order to authenticate it. If the server does not have the same set of LTPA keys that the client does, the decryption fails. The LTPA keys have to be shared between the two cells in order for the server to authenticate an LTPA token sent from the client. The LTPA keys must be shared in the following cases:

- Authentication occurs at the message layer, meaning that client authentication is supported or required on both the client and the server.

- Identity assertion is enabled on both the client and the server, and the server identity is used to establish the trust.

You do not need to share LTPA keys in the following cases:

- Authentication occurs at the transport layer, meaning that client certificate authentication is supported or required on both the client and the server, in addition to the message layer and attribute layer not being supported.
- Identity assertion is enabled on both the client and the server, and an alternative trusted identity is used to establish the trust.

If the client is sending an SSL certificate, the receiving server has to trust the signer of the client certificate. Therefore, you need to add the signer certificate of the client's truststore to the server's truststore. If you skip this step, your client will get an SSLHandshakeException when trying to communicate over SSL with the server.

Cross-cell communication using KRB5

For cross-cell communication using KRB5, the conditions discussed in the previous section for LTPA still apply. In addition, when your client and server are part of two different cells, and these cells do not use the same Kerberos realms, you need to:

1. Set up the Kerberos cross realms as trusted.
2. Add the Kerberos external trusted realm in the CSIv2 inbound/outbound trusted authentication realms.

Cross-registry communication

If you are authenticating to the server with a GSSUP token (user ID and password), you have to ensure that the user ID in the client's registry also exists in the server's registry. If the user ID does not exist on the server side, you need to map the client identity to another identity that does exist. For more information about this mapping, refer to "Identity mapping" on page 220. This additional configuration is necessary in the following cases:

- Authentication occurs at the message layer, meaning that client authentication is supported or required on both the client and the server.
- Identity assertion is enabled on both the client and the server, and an alternative trusted identity and password are used to establish the trust.

Cross-system communication

CSIv2 considerations for cross-system communication are only relevant on the z/OS platform. Certain CSIv2 settings take advantage of a special local communication that is available on z/OS when both the client and the server are on the same system. There are two scenarios in which this local communication is used:

- **Identity assertion:** In order to establish trust between two servers, the sending server sends its server identity. When local communication is used, the server identity of the sending server can be set to the started task ID of the sending server's control region. Note that this scenario only happens when the server identity is not sent or cannot be validated in the message layer (as an LTPA token or GGSUP token), or in the transport layer (as a client certificate).
- **wsadmin client:** When connecting the wsadmin client to a server using the RMI connector, you can set the `loginSource` to `none` and not specify a user ID or password. In this case, the local communication is used to inherit the credentials of the logged-in user and these credentials are used to authenticate to the server.

Scenarios involving these specific security aspects might have worked fine when the client and the server were on the same system, but they will stop working if they are no longer local, which can also explain the difference in behavior between a local client and a remote client.

6.5 Troubleshooting CSIv2

This section describes how to debug error messages and failures related to CSIv2.

6.5.1 Identifying a CSIv2 problem

CSIv2 is the protocol that is used for secure communications between a client and a server over RMI/IIOP. For example, a client invokes a protected method for an EJB that resides on a server. There are typically three areas where you can have a problem:

- **Establishing a connection between the client and the server:** In this case, the request from the client never even reaches the server. This problem can be caused by incompatible CSIv2 settings at the transport layer.
- **Authenticating the client identity:** The request from the client has reached the server. But the identity sent from the client cannot be authenticated, meaning that the server cannot determine the identity, which can be caused by incompatible CSIv2 settings at the message layer or at the attribute layer.
- **Authorizing the client identity:** The request from the client has reached the server, and the client's identity has been authenticated. But the client identity

is not authorized to invoke the EJB method. This problem lies outside the scope of CSIv2 and can be caused by not mapping the client identity to the role required by the method.

6.5.2 Approach to debugging a CSIv2 problem

You can use the following approach when debugging a CSIv2 problem. Note that you do not have to follow this specific order. However, these steps follow an order from a superficial to a more in-depth investigation:

1. Check the logs of both the client and the server for an error message.

   Many times, it is sufficient to simply look at the logs without having to enable any trace. It is important to check both the client's logs and the server's logs. On z/OS, check both the control region and servant region logs of the server.

   If the error message is not clear, find the message ID in the Information Center for additional information and suggested actions. All CSIv2-related messages in WebSphere Application Server start with `JSAS`, and all general security messages start with `SECJ`.

   If you find a CORBA exception with a security minor code (for example, `CORBA.NO_PERMISSION`), refer to "CSIv2 exceptions" on page 222 for more information.

2. Compare the CSIv2 settings of the client and the server.

   Examine the outbound CSIv2 settings of the client (or the sending server), along with the inbound CSIv2 settings of the receiving server. Make sure that they are compatible. For example, if one side has the required setting, the other side must at least be supported.

3. Enable the trace.

   Enable the recommended security and ORB trace on both the client and the server (refer to 6.5.3, "Enabling trace for CSIv2" on page 242). We recommend dynamically enabling this trace immediately before running the test, because it produces a smaller trace, which is easier to debug.

4. Analyze the trace.

   Gather the logs from both the client and the server, along with any ffdc files that were generated. Skim over these logs looking for the phrase `send_exception` near the time of the test failure. This trace entry from the CSIv2 code indicates that an exception occurred and that an exception is being sent back to the originator of the request. From the point where the `send_exception` occurs, search backwards on the same thread for the original exception.

6.5.3 Enabling trace for CSIv2

When enabling trace to debug CSIv2 problems, the trace string to use is the same whether on the client or on the server. However, the way to enable the trace differs. The trace string to enable is:

```
com.ibm.ws.security.*=all:SASRas=all:ORBRas=all
```

Enabling trace on the server

Enabling CSIv2 trace on the server is the same as enabling any other kind of trace. We recommend that you dynamically enable the trace immediately before running the test and that you disable it after the test is complete.

On distributed systems, you can use either the administrative console or wsadmin scripting. On the administrative console, navigate to **Servers** → **Application Servers** → ***server_name*** → **Troubleshooting** → **Change Log Detail Levels**, and select the **Runtime** tab. Enter the trace string here, and then, click **Apply**. For wsadmin scripting, refer to the following article in the Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/txml_aosupport.html

On z/OS, you can additionally use the MVS **`modify`** command to both enable and disable trace.

z/OS: For ease of use, we recommend using the MVS **`modify`** command to enable and disable trace.

For details about the MVS **`modify`** command, refer to this article in the Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rxml_mvsmodify.html

Enabling trace on the client

Enabling trace on the client depends on how the client is being invoked. If you are using launchClient, you need to specify a series of client container name-value pair parameters, in the form of -CC*<name>*=*<value>*:

- -**CCtrace** specifies the trace string.
- -**CCtracefile** specifies the name of the file to which trace is written.
- -**CCtraceMode** indicates the trace format to use.

Example 6-3 on page 243 shows a launchClient command with trace enabled.

Example 6-3 Example of launchClient command with trace

```
launchClient.sh TechnologySamples.ear
-CCtrace=com.ibm.ws.security.*=all:SASRas=all:ORBRas=all
-CCtracefile=myTrace.log -CCtraceMode=basic
-CCjar=BasicCalculatorClient.jar -CCBootstrapPort=2809 add 1 2
```

If you are not using **`launchClient`**, you will need to add a system property to the startup command of the application. This property points to a file containing the trace string and the name of the file where the trace is written. Here are the steps to follow:

1. Copy the `install_root/properties/TraceSettings.properties` file to the classpath of the application client.
2. Edit the `TraceSettings.properties` file. Set the `traceFileName` to the fully qualified name of the output file where the trace is written. Add the recommended trace string for CSIv2. Example 6-4 illustrates the contents of the `TraceSettings.properties` file.

   Example 6-4 Contents of the TraceSettings.properties file

   ```
   traceFileName=/u/bob/myTraceFile.log
   com.ibm.ws.security.*=all:SASRas=all:ORBRas=all
   ```

3. Add the system property `-DtraceSettingsFile` to the **`startup`** command of the client application. Set the value of the property to the name of the properties file containing the trace settings. Example 6-5 shows the command to trace an application program called `com.ibm.samples.MyClienApp`.

   Example 6-5 Calling a client application with trace

   ```
   java -DtraceSettingsFile=TraceSettings.properties
   com.ibm.samples.MyClientApp
   ```

6.5.4 Case studies of CSIv2 problems

The following case studies provide examples of identifying a CSIv2 problem, debugging it, and finding the solution.

Case study 1: Identity assertion

In the scenario, a servlet on server1 invokes a protected method for an EJB on server2. The invocation credential on server1 is bob. Because identity assertion is enabled, the identity of bob gets asserted to server2, which results in the received credential on server2 getting set to `bob`. Furthermore, the trusted identity on server1 has been set to an alternative user ID, `alice`, instead of using the server identity to establish trust between the two servers. Figure 6-4 illustrates the scenario.

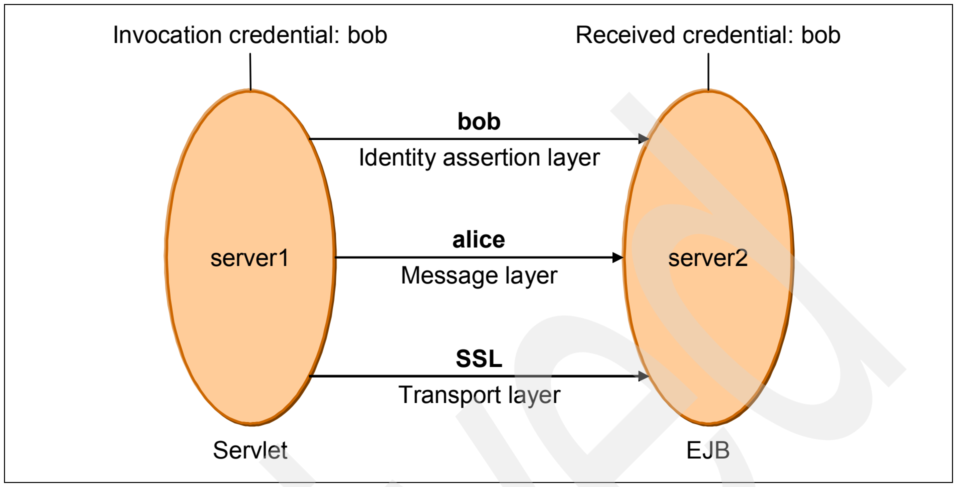


Figure 6-4 Case study 1: Servlet calling an EJB using identity assertion

The problem is that when the servlet on server1 invokes the protected method of the EJB on server2, the application fails.

To analyze, the first step is to examine the logs of the client at the time of the failure. In this case, the client is actually server1. Looking at the systemout log of server1, we find that the server reports a CORBA exception listed in Example 6-6 when trying to invoke the secure EJB on server2.

Example 6-6 Case study 1: Exception on the client

```
org.omg.CORBA.NO_PERMISSION:   vmcid: 0x49424000  minor code: 30D
completed: Maybe
```

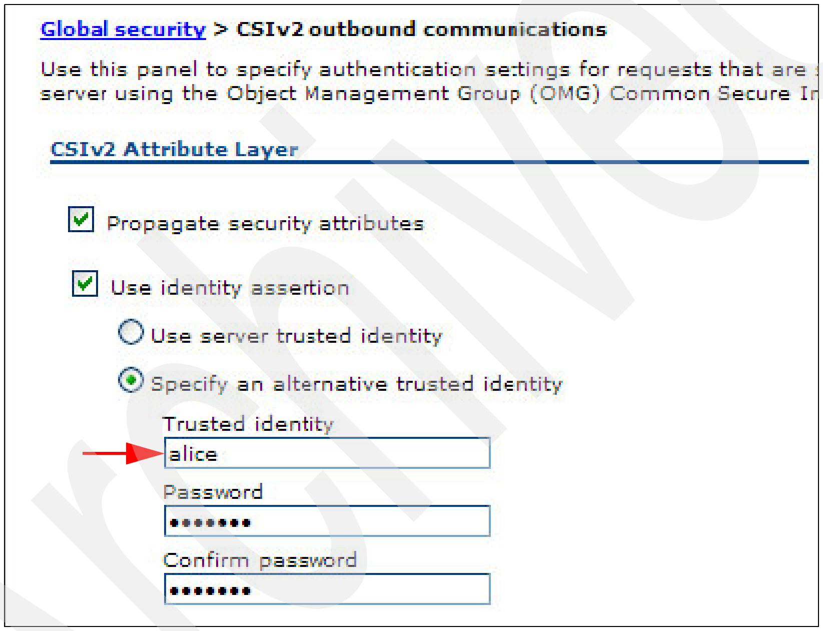
Referring back to “CSIv2 exceptions” on page 222, we can look up the minor code `30D` in the Information Center. We learn that it means that the trusted identity of the sending server is not on the receiving server’s trusted principal list. But we still need more information: What is the trusted identity being sent by server1? What identities does server2 trust? To answer these questions, we can review the logs of the receiving server, where the trust is being established. Looking at the logs of server2 at the time of the failure, we find an error message listed in Example 6-7 on page 245.

Example 6-7 Error message on the server

```
JSAS0499E: The server ID that is received for identity assertion
(alice) does not match any of the configured and trusted server IDs
(jack).
```

The trusted identity sent by server1 is `alice`. The identity that server2 trusts is `jack`.

Note that we can also examine the CSIv2 settings of the servers to get the answers to these questions. We can determine the trusted identity that is being sent by server1 by looking at the outbound CSIv2 settings for server1 in the identity assertion section as shown in Figure 6-5.



Global security > CSIv2 outbound communications

Use this panel to specify authentication settings for requests that are
server using the Object Management Group (OMG) Common Secure In

CSIv2 Attribute Layer

- ✓ Propagate security attributes
- ✓ Use identity assertion
  - ○ Use server trusted identity
  - ● Specify an alternative trusted identity

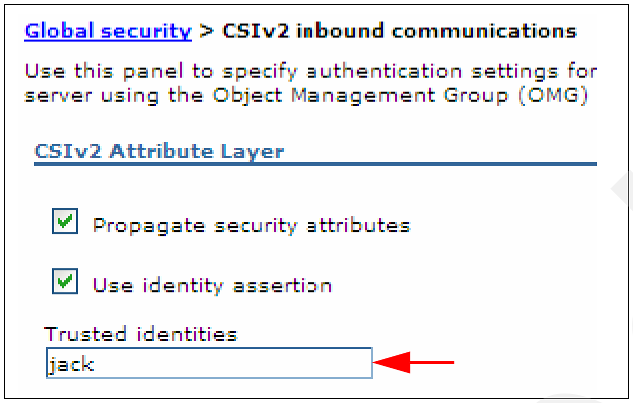
Trusted identity: alice

Password: •••••••

Confirm password: •••••••

Figure 6-5 Case study 1: CSIv2 outbound settings on the sending server

To determine which identities are trusted by the receiving server, we examine the inbound CSIv2 settings of server2 as illustrated in Figure 6-6 on page 246.


Global security > CSIv2 inbound communications

Use this panel to specify authentication settings for server using the Object Management Group (OMG)

CSIv2 Attribute Layer

[✓] Propagate security attributes

[✓] Use identity assertion

Trusted identities
jack

Figure 6-6   Case study 1: CSIv2 inbound setting on the receiving server

In conclusion, the problem is that server2 does not trust `alice`.

For the solution, there are two possible fixes to this problem. You can modify the sending server server1's trusted identity to `jack`, which is already trusted by the receiving server. Or, you can modify the receiving server server2's list of trusted identities by adding `alice` to this list.

Case study 2: Client authentication

In this scenario, two security domains are defined: `ldapDomain` and `localOSdomain`. Server1 is mapped to ldapDomain, and server2 is mapped to localOSdomain. A servlet on server1 invokes a protected method of an EJB on server2. The sending server authenticates to the receiving server with a GSSUP token with user id `bob`. This scenario is depicted in Figure 6-7 on page 247.

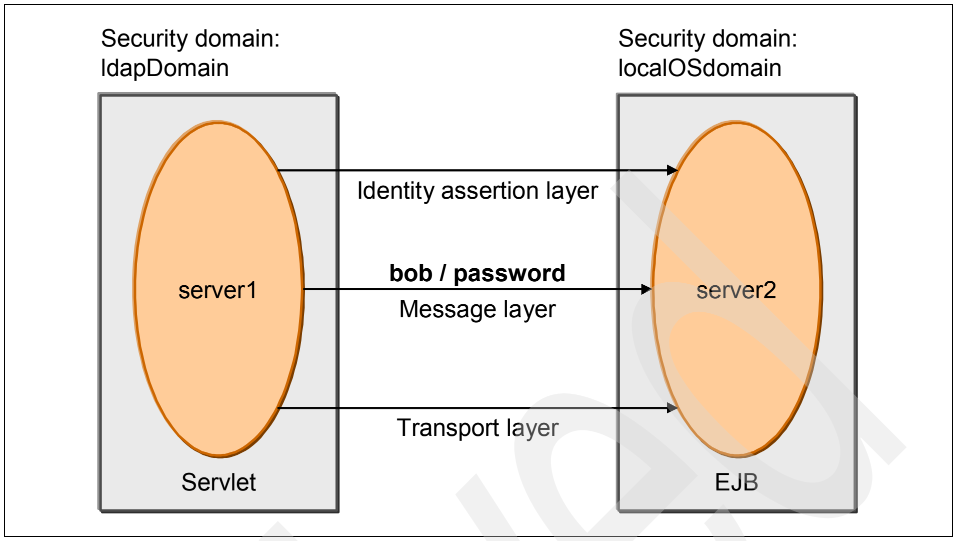


Figure 6-7 Case study 2: Servlet calling an EJB using client certificate authentication

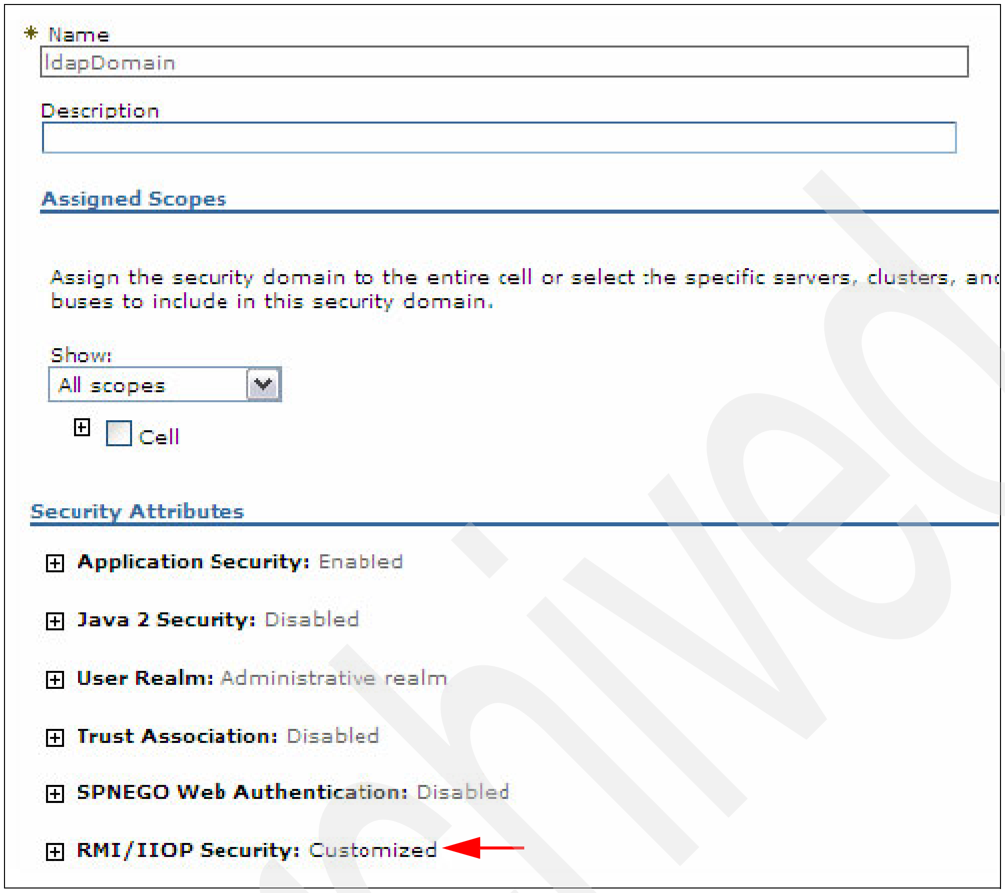
The problem is that the servlet on server1 receives an error message when trying to call the protected method of the EJB on server2.

To analyze it, the first step is to examine the logs of the client, server1, at the time of the failure. We find a CORBA exception listed in Example 6-8.

Example 6-8 Case study 2: Exception on the client

```
org.omg.CORBA.NO_PERMISSION: vmcid: 0x49421000 minor code: 92
ERROR: Unauthenticated credential found, client auth required by client
or server, throwing NO_PERMISSION.
```

The minor code `92` means that a credential is not available when it is required. In this case, we want to use client authentication, so we need to examine why the sending server is sending an unauthenticated credential, instead of `bob`. We need to examine the outbound CSIv2 settings of server1. However, because server1 is part of the security domain ldapDomain, it is possible these settings are different from those settings defined in the global security. A quick way to analyze it is to navigate to the security domain to check the RMI/IIOP Security setting for either `Global security settings` or `Customized`. As shown in Figure 6-8 on page 248, the ldapDomain has its own CSIv2 settings.


* Name
ldapDomain

Description

Assigned Scopes

Assign the security domain to the entire cell or select the specific servers, clusters, and
buses to include in this security domain.

Show:
All scopes

Cell

Security Attributes

Application Security: Enabled

Java 2 Security: Disabled

User Realm: Administrative realm

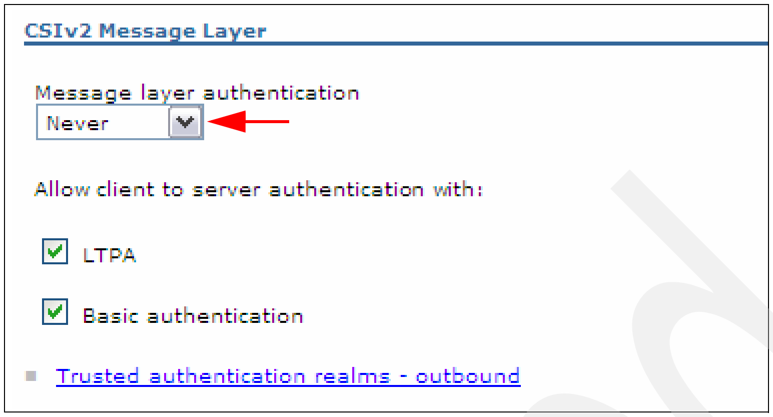
Trust Association: Disabled

SPNEGO Web Authentication: Disabled

RMI/IIOP Security: Customized

Figure 6-8 Case study 2: Security attributes for ldapDomain

We examine the outbound CSIv2 settings of the security domain ldapDomain. As shown in Figure 6-9 on page 249, we observe that the sending server is set to never authenticate using the message layer.


CSIv2 Message Layer

Message layer authentication
Never

Allow client to server authentication with:

✓ LTPA

✓ Basic authentication

Trusted authentication realms - outbound

Figure 6-9 Case study 2: CSIv2 outbound settings for client

This setting clearly conflicts with the intent of the user scenario to authenticate on the message layer with a GSSUP token.

In conclusion, the problem is that the receiving server requires client authentication, but the sending server does not support it.

To solve this problem, the outbound CSIv2 settings for the security domain ldapDomain must be modified. Client authentication at the message layer must be set to at least `supported`.

6.6 References

You might find the following references useful:

- IBM Education Assistant for WebSphere Application Server V6 Security: CSIv2

  `http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v6/was/6.0/Security/WASv6_Sec_CSIv2/player.html`
- Object Management Group (OMG) Security Web site, CSIv2 topic:

  `http://www.omg.org/technology/documents/formal/omg_security.htm#CSIv2`
- *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316
- The Common Object Request Broker: Architecture and Specification:

  `http://www.omg.org/docs/formal/98-12-01.pdf`

Part 2

Application security

7

Application security

Application security provides application isolation and the requirements for authenticating users of the applications. Applications can be secured in a declarative manner or programmatically.

This chapter provides a look at how security is applied when applications are deployed. It contains the following chapters:

7.1 Application security design considerations

Application security implementation requires close interaction and planning among application developers, security specialists, and administrators. Two primary decisions must be addressed during the planning stage for the overall application security design in an enterprise.

7.1.1 Programmatic and declarative security

Management of security in an enterprise is made simpler if all application development uses the same approach to security.

Using programmatic security means that the security policies are enforced in the application using the Java Platform, Enterprise Edition (JEE) security application programming interface (API).

Using declarative security means that security policies are configured independently of the logic of the application code. Security information is specified using metadata, either through annotations in the application or in the EJB deployment descriptor. Security is enforced by the application server.

When *declarative security* is used by any application deployed in the application server, application security has to be enabled for the run time. However, if applications rely only on *programmatic security*, administrators do not necessarily have to enable application security.

More information about securing applications using programmatic or declarative security can be found in Chapter 9, "Securing an Enterprise JavaBeans application" on page 325 and Chapter 8, "Securing a Web application" on page 267.

In either case, security polices are role-based and roles must be mapped to specific users or groups. This mapping can be done at application deployment, after the application is deployed using the administrative tools, or in the development environment through the application bindings deployment descriptor.

7.2 Deploying a secured enterprise application

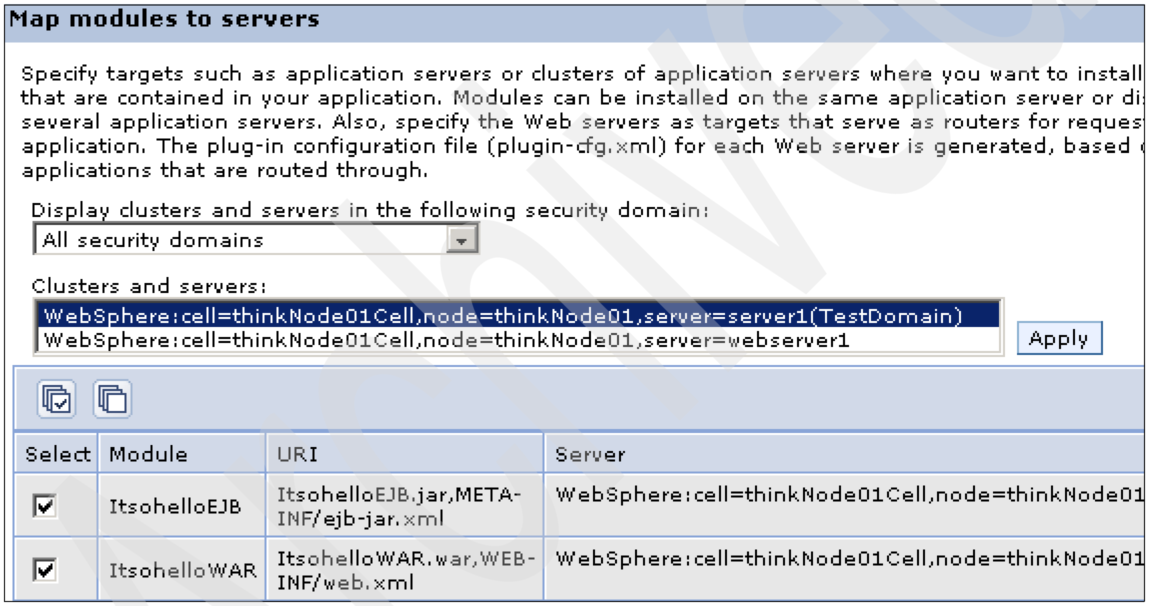
Deploying a secured application is similar to deploying a non-secured enterprise application. The only difference is that during deployment, administrators can perform the role mapping for users and groups, as well as the run-as mapping.

This section discusses deployment and post-deployment considerations for managing application security.

7.2.1 Mapping modules to servers filtered by security domains

When deploying an enterprise application, administrators must select a deployment target for each application module (Figure 7-1). In a single server environment, the available targets are application servers and Web servers. In a clustered environment, you also can select clusters.

In an environment with multiple security domains, administrators need to take the security domain configurations into consideration when choosing the deployment target for the modules.



Map modules to servers

Specify targets such as application servers or clusters of application servers where you want to install that are contained in your application. Modules can be installed on the same application server or di several application servers. Also, specify the Web servers as targets that serve as routers for reques application. The plug-in configuration file (plugin-cfg.xml) for each Web server is generated, based applications that are routed through.

Display clusters and servers in the following security domain:
All security domains

Clusters and servers:
WebSphere:cell=thinkNode01Cell,node=thinkNode01,server=server1(TestDomain)
WebSphere:cell=thinkNode01Cell,node=thinkNode01,server=webserver1

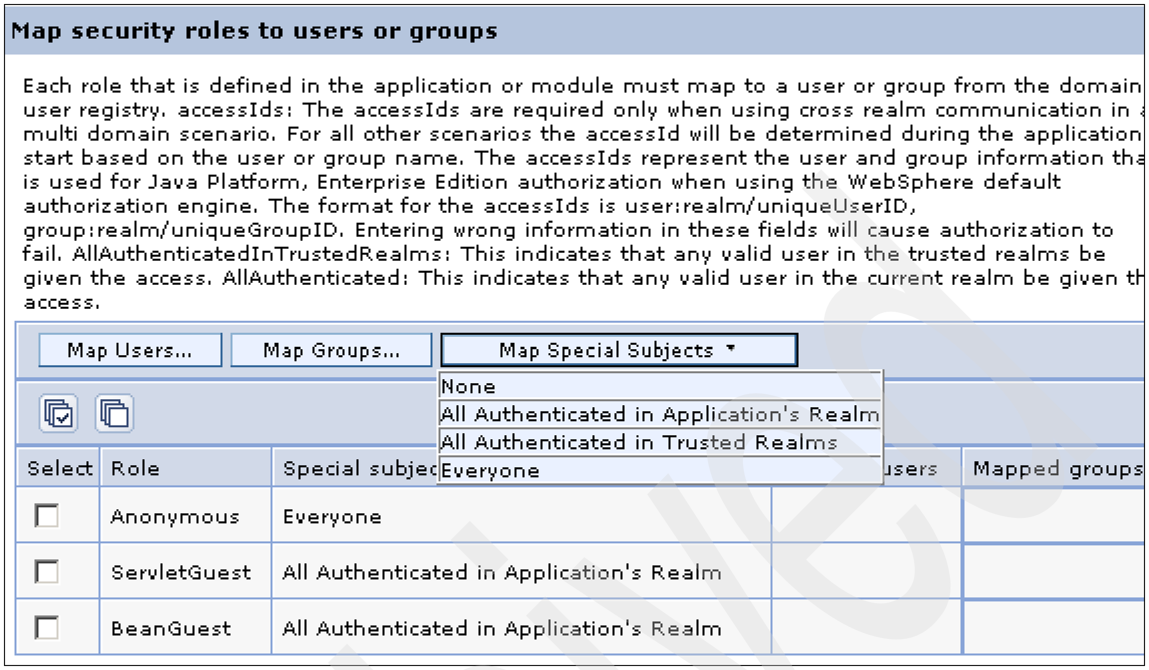
Apply

Select	Module	URI	Server
✓	ItsohelloEJB	ItsohelloEJB.jar,META-INF/ejb-jar.xml	WebSphere:cell=thinkNode01Cell,node=thinkNode01
✓	ItsohelloWAR	ItsohelloWAR.war,WEB-INF/web.xml	WebSphere:cell=thinkNode01Cell,node=thinkNode01

Figure 7-1 Deploying a secure application: Map modules to servers

7.2.2 Role mapping during application installation

Another step to consider during application deployment is to *map security roles to users or groups* (Figure 7-2 on page 256). At this step, administrators have the option of selecting any of the roles and assigning a user or a group from the user registry using one of the lookups. Administrators can also assign one of the special subjects to the role.


Map security roles to users or groups

Each role that is defined in the application or module must map to a user or group from the domain user registry. accessIds: The accessIds are required only when using cross realm communication in a multi domain scenario. For all other scenarios the accessId will be determined during the application start based on the user or group name. The accessIds represent the user and group information tha is used for Java Platform, Enterprise Edition authorization when using the WebSphere default authorization engine. The format for the accessIds is user:realm/uniqueUserID, group:realm/uniqueGroupID. Entering wrong information in these fields will cause authorization to fail. AllAuthenticatedInTrustedRealms: This indicates that any valid user in the trusted realms be given the access. AllAuthenticated: This indicates that any valid user in the current realm be given th access.

Map Users...
Map Groups...
Map Special Subjects
None
All Authenticated in Application's Realm
All Authenticated in Trusted Realms
Everyone

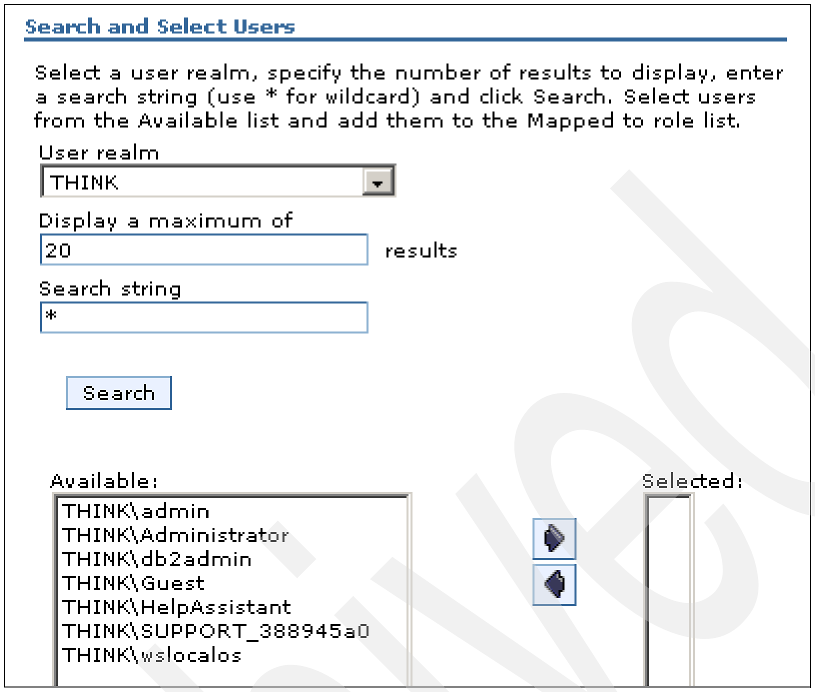
Select	Role	Special subjec	users	Mapped groups
☐	Anonymous	Everyone		
☐	ServletGuest	All Authenticated in Application's Realm		
☐	BeanGuest	All Authenticated in Application's Realm		

Figure 7-2 Deploying a secure application: Map security roles to users or groups

To use a special subject, select one of the following options from the Map Special Subjects drop-down list:

- None: No mapping to the role is performed.
- All Authenticated in Application’s Realm: All authenticated users in the application’s realm are mapped to this role.
- All Authenticated in Trusted Realms: All of the users in the trusted realms. If trusted realms are configured, a drop-down list of realms to search is displayed. Users from the non-default realm are displayed as user@realm. Everyone: Everyone is mapped to this role (essentially, there is no security).

Alternatively, specific users or groups can be mapped by clicking the corresponding **Map Users** or **Map Groups**. When you perform this mapping, a user registry search is available to aid the mappings of users and groups (Figure 7-3 on page 257).


Search and Select Users

Select a user realm, specify the number of results to display, enter a search string (use * for wildcard) and click Search. Select users from the Available list and add them to the Mapped to role list.

User realm: THINK

Display a maximum of: 20 results

Search string: *

Available:

THINK\admin
THINK\Administrator
THINK\db2admin
THINK\Guest
THINK\HelpAssistant
THINK\SUPPORT_388945a0
THINK\wslocalos

Selected:

Figure 7-3 Deploying a secure application: Map users and groups to roles

Test the role mapping

After the application is installed and started, a Java 2 Platform, Enterprise Edition (J2EE) client can be run to test the secured bean. In the command window, a realm prompt window will appear with the current realm filled in. A valid realm user and password are required to access the protected bean.

Example 7-1 shows a sample of the client script file.

Example 7-1 Client script file

```
@echo off

setlocal

rem ************************************
rem  Modify this block according to the
rem  setup of your system
rem ************************************
set WAS_HOME=C:\IBM\WebSphere\AppServer
set SERVER_HOST=think
set SERVER_PORT=2809
```

```
%WAS_HOME%\bin\launchclient.bat ItsohelloEAR.ear
-CCBootstrapHost=%SERVER_HOST% -CCBootstrapPort=%SERVER_PORT%

endlocal
```

Note: Substitute the environment parameter with values from your own testing environment.

If you execute this script in a command window, you will see output similar to Example 7-2. Select option **b** to test.

Example 7-2 Executing client script

```
IBM WebSphere Application Server, Release 7.0
Java EE Application Client Tool
Copyright IBM Corp., 1997-2008
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the Java EE Application Client Environment.
[3/23/09 22:40:17:885 EDT] 00000000  W UOW=null
source=com.ibm.ws.ssl.config.SSL
Config org=IBM prod=WebSphere component=Application Server
thread=[P=417619:O=0:
CT]
          CWPKI0041W: One or more key stores are using the default
password.
WSCL0035I: Initialization of the Java EE Application Client Environment
has comp
leted.
WSCL0014I: Invoking the Application Client class
com.ibm.itsohello.j2eeclient.J2
EEClient



J2EE Itsohello clients:


a. UNSECURED CLIENT.
   Access the unsecured Hello bean. If you still get an authentication
   challenge window, just click "Cancel". Or you can also change the
   property
   "com.ibm.CORBA.loginSource" to "none" in the file "sas.client.props"
   found in your WebSphere application runtime client.

b. SECURED CLIENT.
```

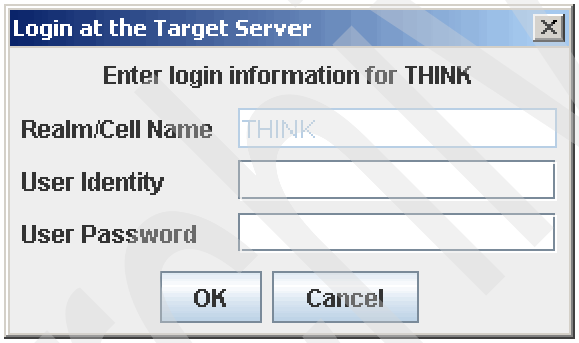
```
   Access the secured Hello bean. You must be authenticated; otherwise,
the app will throw an exception. If you do not get an authentication
challenge window, you need to change the property
"com.ibm.CORBA.loginSource" to "prompt" in the file "sas.client.props"
found in your WebSphere application runtime client.

c. SECURED CLIENT with JAAS.
   Access the secured Hello bean using JAAS. Authentication is done via
JAAS.

d. SECURED CLIENT with JAAS using custom callback handler.
   Similar like (c) but with custom callback handler.

Please enter your choice (a/b/c/d): b
```

Selecting this option causes the login window (Figure 7-4) to appear.


Login at the Target Server

Enter login information for THINK

Realm/Cell Name: THINK

User Identity:

User Password:

Figure 7-4 Deploying a secure application: J2EE client test

Use `wslocalos/wslocalos` as the username/password pair. The result in Example 7-3 will be displayed in the command window.

Example 7-3 Results from the test

```
Accessing SecuredHello bean
Message from Hello bean: [Secured] Hello to you wslocalos (roles:
Anonymous Bean
Guest)
```

7.2.3 Run-As role mappings

If the application has Enterprise JavaBeans (EJBs) or EJB methods with Run-As role mappings, the installation process includes the step: *Map RunAs roles to users,* which allows you to specifically assign a user name and password (an identity) to a Run-As (delegation) definition.

If the application has EJB or EJB methods with Run-As system mappings, the installation process includes the step: *Correct use of system identity*, which gives you the opportunity to override the default system identity with a specific user mapping.

For more information about Run-As mapping, refer to 9.6, “Delegation” on page 353.

7.2.4 Unprotected 2.x methods

If the application has EJB methods without security assignments, the installation process includes the step: *Ensure all unprotected 2.x methods have the correct level of protection*. This step gives you the opportunity to assign a role to these methods on a per EJB basis (not on a per method basis). You can also exclude the methods so that they cannot be accessed, or you can clear the check mark from them so that they can be accessed by everyone.

For more information about EJB method security, refer to 9.1, “Application security” on page 327.

7.2.5 Mapping roles at assembly compared to deployment

Role mappings can also be defined in the enterprise archive during assembly time, just before deployment using Rational® Application Developer. Even if the mappings have been performed previously, administrators can review and modify them during deployment or even later, as discussed in 7.3, “Role mapping after installation” on page 260.

7.3 Role mapping after installation

Role mappings can be changed after installation:

1. In the administrative console, select **Applications** → **Application types** → **WebSphere enterprise applications** from the menu.
2. Select the application.

You will find the following items under the Detail Properties section:

– Security role to user/group mapping

– User Run-As roles

  Selecting each of these options will open the same configuration page that you see during deployment.

> **Note:** The *User Run-As roles* configuration link only opens in the administrative console when the application uses Run-As delegation. WebSphere detects whether this configuration exists in the application and changes the interface accordingly.
>
> The *Ensure all unprotected 2.x methods have the correct level of protection* configuration is not available after deployment. After the methods are defined as unchecked, excluded or mapped to a role, this configuration does not change.

3. Make the appropriate mappings and save the configuration.
4. Restart the Enterprise Application for the changes to become effective.

7.4 Mapping roles in the development environment

Mapping roles in Rational Application Developer for WebSphere Software V7.5 is done in the application bindings deployment descriptor (`ibm-application-bnd.xml`).

By default, JEE 5 applications do not have deployment descriptors, so you need to generate one. The quickest way is to select the enterprise application project, right-click, and then select **Java EE** → **Generate WebSphere Bindings Deployment Descriptor**. Example 7-4 shows the generated application bindings deployment descriptor.

Example 7-4 Generated application bindings deployment descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<application-bnd
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee
http://websphere.ibm.com/xml/ns/javaee/ibm-application-bnd_1_0.xsd"
   version="1.0">
</application-bnd>
```

After the descriptor is generated, follow these steps to add and map the security roles:

1. In the Enterprise Explorer view, expand the enterprise application project and `META-INF` folder.
2. Double-click **ibm-application-bnd.xml** to open the Application Bindings Editor. Select the **Design** tab.
3. Select **Application Bindings** and click **Add**.
4. Select the **Security Role** element as shown in Figure 7-5, and click **OK**.

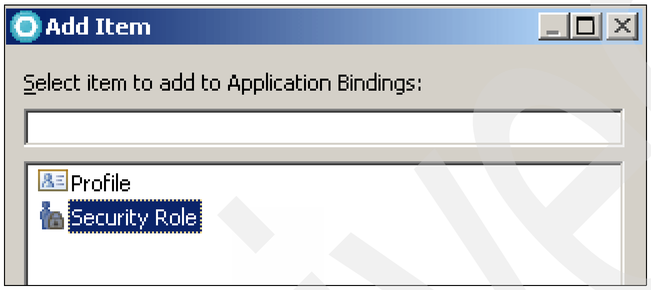
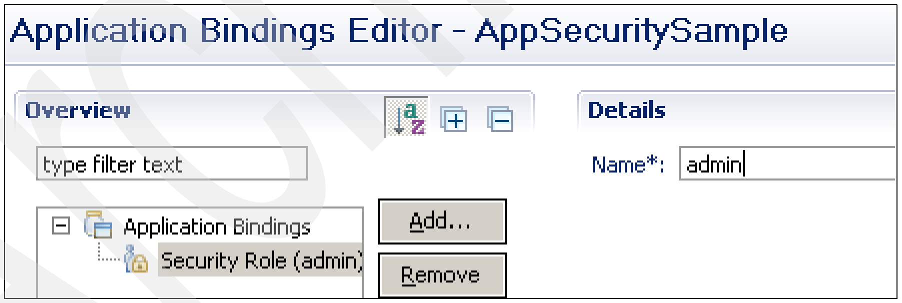


Figure 7-5 Adding Security role

5. In the binding editor, select the new Security role element and enter the role name (`admin` in this example). Refer to Figure 7-6.



Name*: admin

Figure 7-6 Defining the security role

6. Repeat steps 3-5 to add additional roles. In this example, the `user` role is added.
7. The next step is to select each role and map users, groups, or special subjects to it.

   You can map security roles to the following objects:

   – Group: A group defined in the user registry

- User: A user ID defined in the user registry
- Special Subject: One of the following special subjects:
  - Everyone: Represents all users, even unauthenticated. Anyone can access resources protected by this role.
  - All Authenticated Users (All authenticated in the application's realm): Represents all valid users from the application realm. When you map this subject to a role, all users that have successfully authenticated can access resources protected by this role.
  - All Authenticated in Trusted Realms: Represents all valid users from all trusted realms. This option is only available if multiple security domains are defined.

Follow these steps:

a. Select the first role, **Security role (admin)**, and click **Add**.

b. For this example, select **User** as the type of element that you will map (Figure 7-7), and click **OK**.

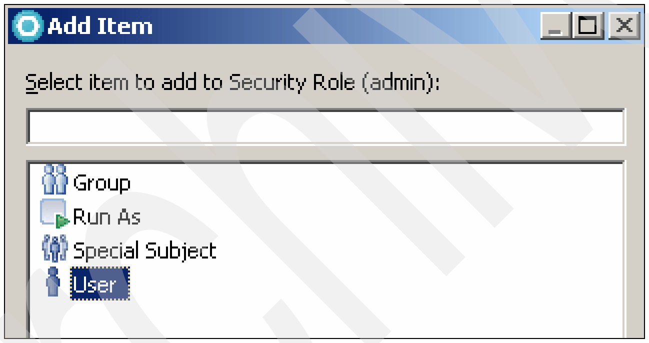


Figure 7-7 Mapping role to user

8. Select the new User element and enter the user ID from the application server user registry in the Name field as shown in Figure 7-8 on page 264

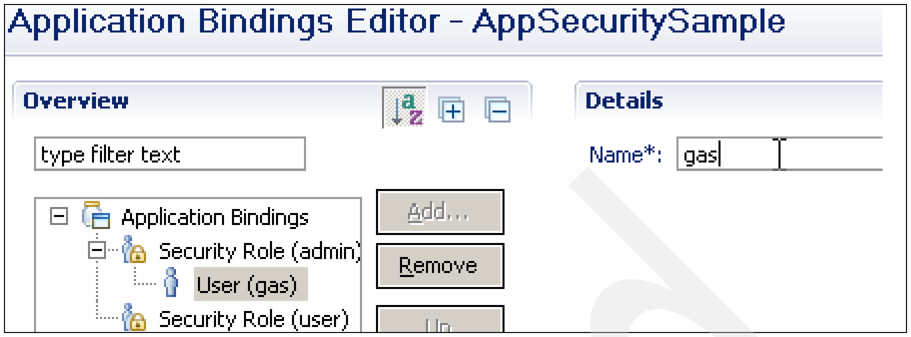


Figure 7-8 Specifying user ID for security role

Next, map the user role by selecting **Security role (user)**, and then, clicking **Add**.

9. Select **Special Subject** as the mapping element (Figure 7-9), and click **OK**.

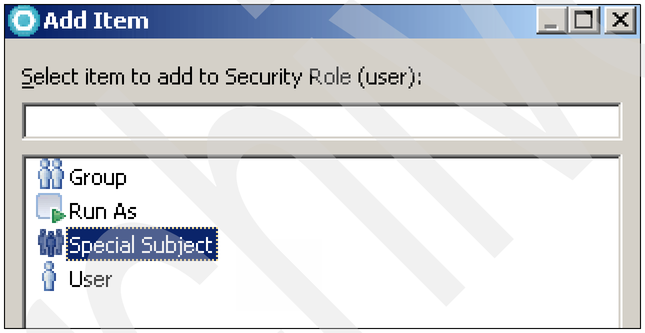


Figure 7-9 Mapping role to Special subject

10. Select the new **Special Subject** element and select **All Authenticated Users** as shown in Figure 7-10 on page 265.

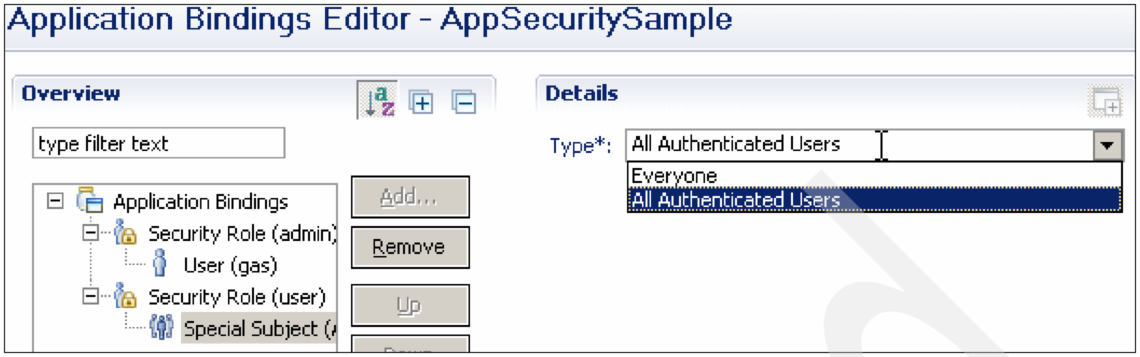
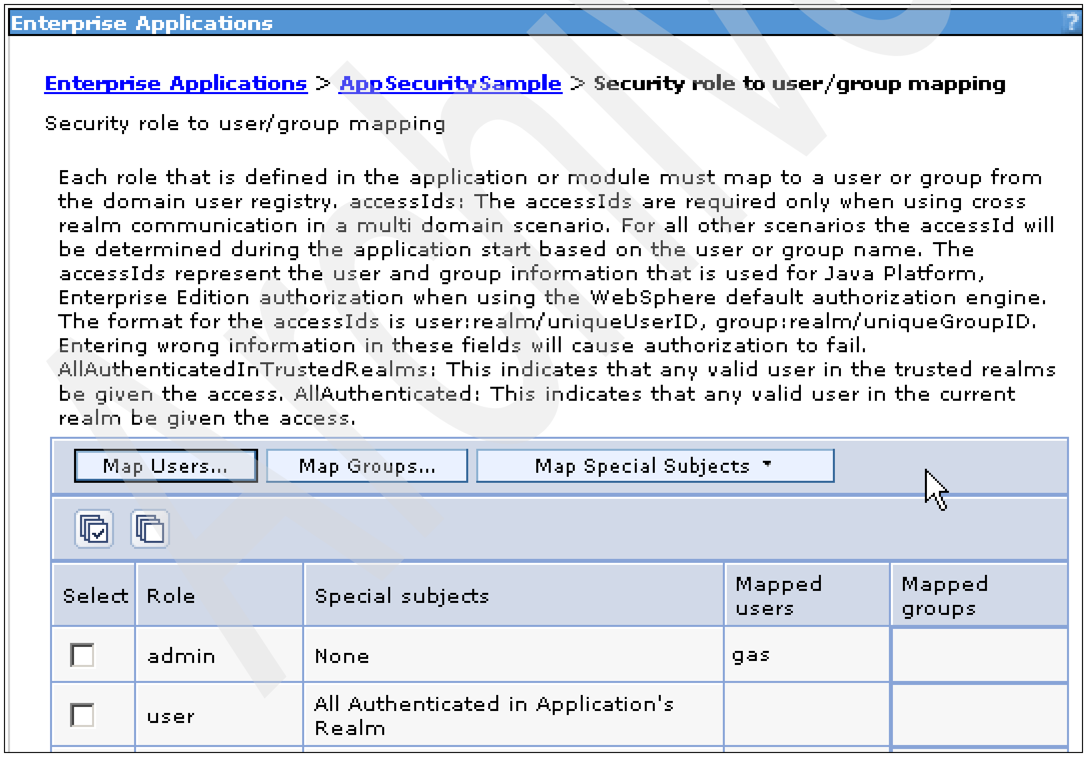


Figure 7-10 Specifying Special Subject

11.Save the binding file.

After the deployment of the application to the server, you can view or edit security role mappings in the administrative console as shown in Figure 7-11.


Enterprise Applications

Enterprise Applications > AppSecuritySample > Security role to user/group mapping

Security role to user/group mapping

Each role that is defined in the application or module must map to a user or group from the domain user registry. accessIds: The accessIds are required only when using cross realm communication in a multi domain scenario. For all other scenarios the accessId will be determined during the application start based on the user or group name. The accessIds represent the user and group information that is used for Java Platform, Enterprise Edition authorization when using the WebSphere default authorization engine. The format for the accessIds is user:realm/uniqueUserID, group:realm/uniqueGroupID. Entering wrong information in these fields will cause authorization to fail. AllAuthenticatedInTrustedRealms: This indicates that any valid user in the trusted realms be given the access. AllAuthenticated: This indicates that any valid user in the current realm be given the access.

Select	Role	Special subjects	Mapped users	Mapped groups
☐	admin	None	gas	
☐	user	All Authenticated in Application's Realm		

Figure 7-11 Security role mapping through the admin console

Test environment on Rational Application Developer: To be able to use the administrative console of the WebSphere Application Server V7 test environment server to map users and groups to roles that are defined by annotations, you have to change the publishing settings to run with resources on the server.

In Rational Application Developer V7.5, in the **Servers** view, double-click **WebSphere Application Server V7.0 at localhost**. In the “Publishing settings for WebSphere Application Server” section, select **Run server with resources on Server**.

8

Securing a Web application

This chapter discusses various aspects of securing Web applications. It shows how to protect access to Web application resources using declarative and programmatic security. It presents authentication mechanisms and explains in detail how to configure form-based and client certificate authentication.

If you need to implement more sophisticated authentication, we describe several WebSphere Application Server security APIs that can be used to customize the login process.

This chapter assumes basic Java Platform, Enterprise Edition (JEE) knowledge, so we do not explain terms, such as HTML pages, JavaServer™ Page (JSP™), and servlets in detail.

This chapter contains the following topics:

8.1 Application security

A typical JEE application consists of both Web and EJB modules. Web components build the user interface to the application business logic. A Web application can be accessed over the network by many users that perform different roles and require access to different functions. It is vital that you properly secure access to application components.

There are two primary aspects of security in Web applications: authentication, which provides information about who the user is; and authorization, which determines the resources that are available to the user.

WebSphere Application Server supports the authentication mechanisms that are required by the JEE specification, including HTTP basic and form-based or client certificates. In addition, it offers Kerberos support and support for third-party authentication proxies.

After a user is authenticated, that user must be authorized to access protected resources. JEE offers role-based authorization, which grants access to the resource based on the user's security role. Web applications can use one or both mechanisms offered by JEE: declarative security that allows you to define security constraints via a deployment descriptor and programmatic security that allows you to embed more fine-grained access decisions inside a servlet or JSP page.

When security requirements are so complex that JEE security is not enough, consider using a third-party security framework or an external Java Authorization Contract for Containers (JACC) provider. These concepts are beyond the scope of this book and are not discussed.

When you design an application, always consider using JEE security first rather than a custom-developed framework. JEE security is built on a strong security infrastructure that is already in place and enforced by the application server.

8.2 Declarative security

Declarative security defines security information (security roles, access constraints, and authentication requirements) in the external XML Web deployment descriptor file. There are no security API calls hard-coded in the servlets or JSPs. Security is enforced by the Web container based on the requirements that are specified by the descriptor.

One of the benefits of declarative security is the ability to change an application's security settings according to client needs without changing the application code. For example, you can opt to use client certificate authentication instead of HTTP basic by simply changing the Web deployment descriptor. All deployment descriptors can be created and modified using Rational Application Developer for WebSphere Software V7.5.

JEE uses a role-based security model, meaning access to application resources is granted based on the security role. The *security role* is a logical grouping of principals.

Terms:

- A *principal* can be authenticated, usually a user.
- A *role* is a logical group of principals that provides a set of permissions. Access to operations is controlled by granting access to a role.
- A *security constraint* is a declaration of how to protect a resource.

Using declarative security involves the following steps:

1. Define security roles for the application. This step simply identifies roles that describe the type of activities.
2. Define security constraints that define what resources are protected, the roles to which a user has to be assigned in order to access those protected resources, and constraints on data transport (using Secure Sockets Layer (SSL)).
3. Deploy the application and map the security roles to actual users and groups from the application server security realm.

8.2.1 Defining security roles for an application

The following steps describe how to create a security role using Rational Application Developer:

1. In the Enterprise Explorer view of the Java EE perspective, expand the Web module and double-click Web deployment descriptor to open it.

   No deployment descriptor: If there is no descriptor, you can generate a deployment descriptor by right-clicking the project name and selecting **Java EE → Generate Deployment Descriptor Stub**.

   Select the **Design** tab.

2. Select the **Web Application** element, and click **Add**.
3. Select the **Security Role** element as shown in Figure 8-1, and click **OK**.

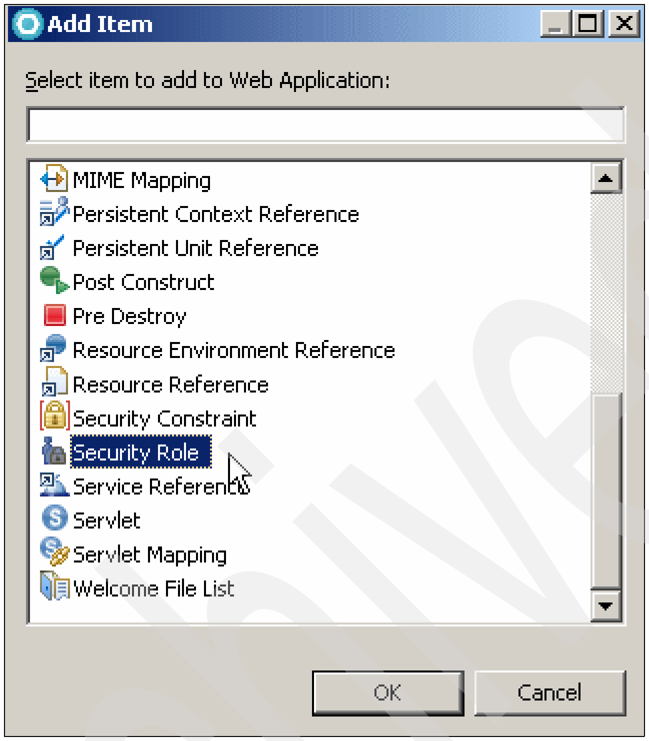
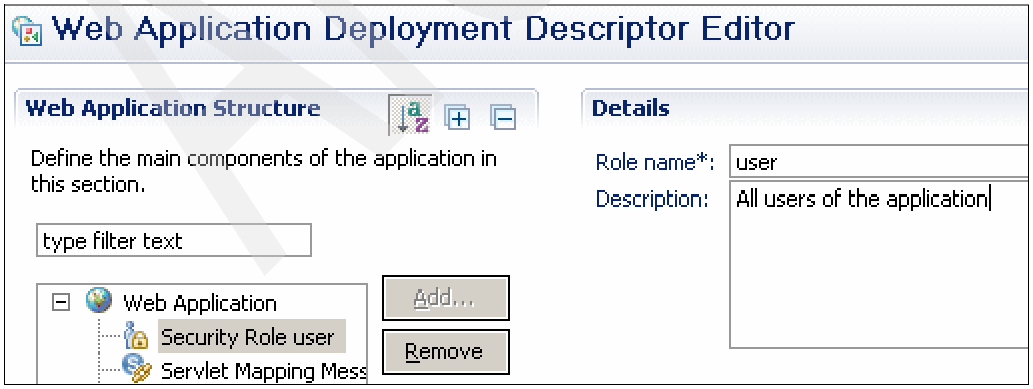


Figure 8-1 Adding Security Role element

4. Select the new Security Role element and enter the role name and description as shown in Figure 8-2. In this example, `user` is the role name.


Web Application Deployment Descriptor Editor
Web Application Structure
Define the main components of the application in this section.
type filter text
Web Application
Security Role user
Servlet Mapping Mess
Add...
Remove
Details

Role name*: user
Description: All users of the application

Figure 8-2 Editing the security role

5. Repeat steps 3-5 to add additional roles. In this example, we added the `admin` security role.
6. Save the file.

The new <*security-role*> elements are added to the Web deployment descriptor. Select the **Source** tab in the editor to see the entire XML file (refer to Example 8-1).

Example 8-1 Security role element in the Web deployment descriptor

```
<security-role>
      <description>All users of the application</description>
      <role-name>user</role-name>
   </security-role>
   <security-role>
      <description>Administrators</description>
      <role-name>admin</role-name>
</security-role>
```

8.2.2 Defining security constraints

Security constraints are a declarative way of specifying access permissions to Web resources. A constraint consists of the following elements:

- Web resource collections
- Authorization constraints
- User data constraints

Web resource collection

A *Web resource collection* is a list of URL patterns that identifies protected resources and the list of HTTP methods that are used to access those resources.

A *URL pattern* is a URI that is relative to the application context. Patterns can include:

- Path mapping, starting with "/" and ending with "/*"

  This pattern identifies any resource that starts with a given path, for example, `/catalog/*` or `/europe/poland/*`

- Extension mapping, starting with "*."

  This pattern identifies any resource with the given extension, for example, `*.jsp` or `*.gif`

- Default servlet mapping, containing only "/"

  This pattern identifies the default servlet of the application.

- Exact matches

  This pattern uses a string that represents a specific resource, for example, `/snoop` is a servlet mapping and `/list/banner.jsp` is a file mapping.

If the same resource matches several URL patterns at run time, the Web container uses the following rules to determine which security constraint to use. The URL is matched in the following order:

1. Exact match of the request path
2. The longest path prefix match
3. The extension mapping match
4. If the same URL pattern and HTTP method occur, the result is the union of authorization constraints. The exception is authorization constraints with no roles defined, which overrides other constraints and denies access to the resource.

HTTP method names are no longer restricted to standard methods, such as GET, POST, HEAD, TRACE, OPTIONS, PUT, and DELETE, but they can be any name that does not contain a control character or separator. For example, HTTP method names can be WEBDAV extension methods, such as LOCK, UNLOCK, COPY, and MOVE.

HTTP methods: If no HTTP methods are specified, the security constraint applies to all HTTP methods.

Authorization constraints

An *authorization constraint* defines which roles are authorized to access resources specified by Web resource collection. If there is no authorization constraint, anyone can access the resources. If an authorization constraint is defined without any roles, access to those resources is prohibited.

User data constraints

A *user data constraint* determines requirements for the transport layer security. This constraint can use the following transport guarantee settings:

- `CONFIDENTIAL`: Defines that the data sent over the network must be encrypted
- `INTEGRAL`: Defines that the data cannot be changed in transit
- `NONE`: Indicates that there are no restrictions on the transport and that the container must accept requests even on an unprotected channel

Setting the transport guarantee to `CONFIDENTIAL` or `INTEGRAL` requires the usage of the SSL (https).

Best practice: If an application sends sensitive data, for example, login names, credit card numbers, and so forth, we recommend that the transport guarantee is set to `CONFIDENTIAL`. The Web container will ensure that the application can only be accessed via secured connection.

Configuring the constraints

The following steps describe how to create security constraints using Rational Application Developer:

1. In the Enterprise Explorer view of the Java EE perspective, expand the Web module and double-click Web deployment descriptor to open it.

   Select the **Design** tab.

2. Select **Web Application**, and click **Add**.
3. Select **Security Constraint** as shown in Figure 8-3, and click **OK**.

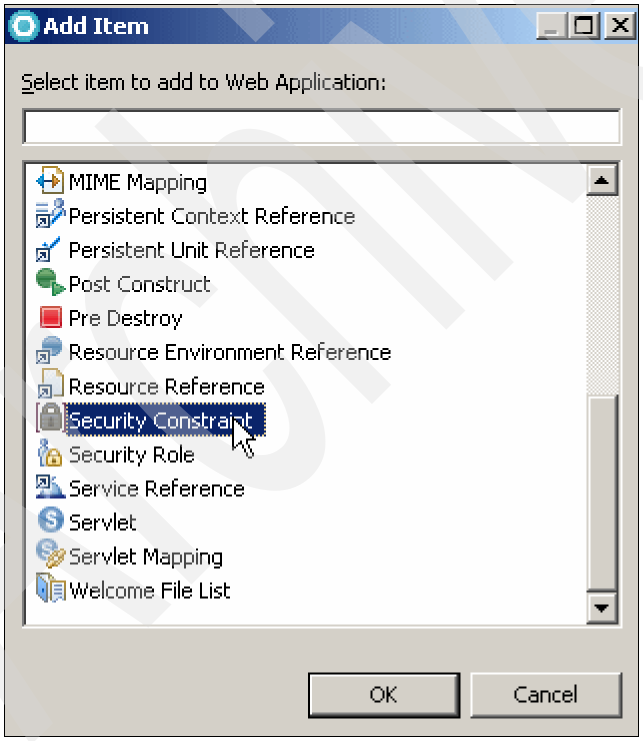


Figure 8-3 Adding a security constraint

4. Select **Web Resource Collection** under the new security constraint element and specify the resources and methods to protect. In this example:

   – HTTP Method list: `GET`, `POST`, and `HEAD`
   – URL Pattern list: `/*`
   – Web Resource Name: `All resources`

   Refer to Figure 8-4.

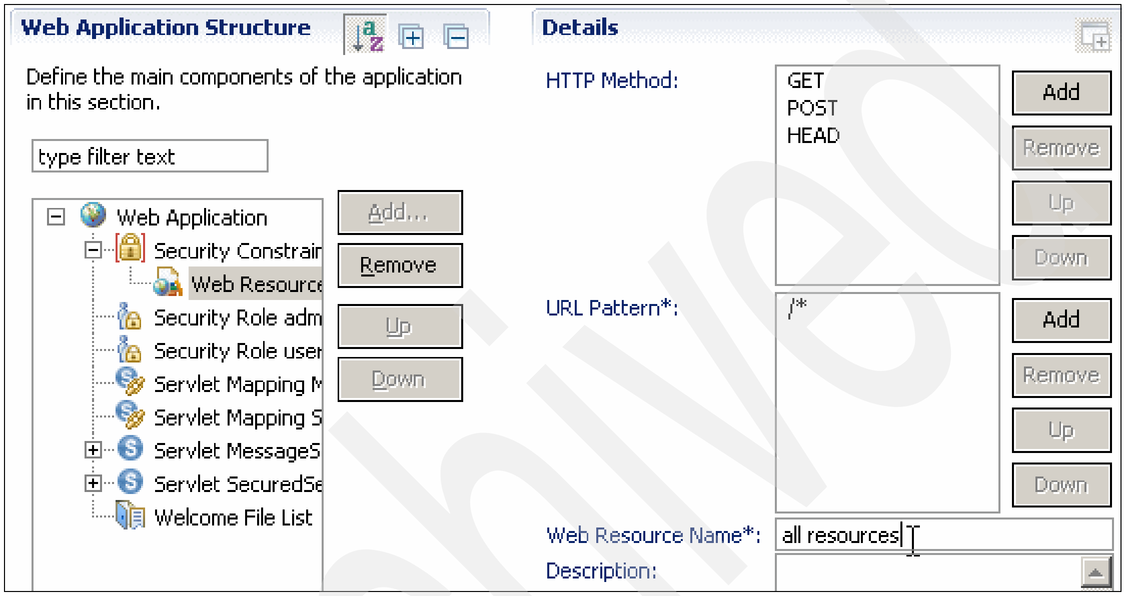
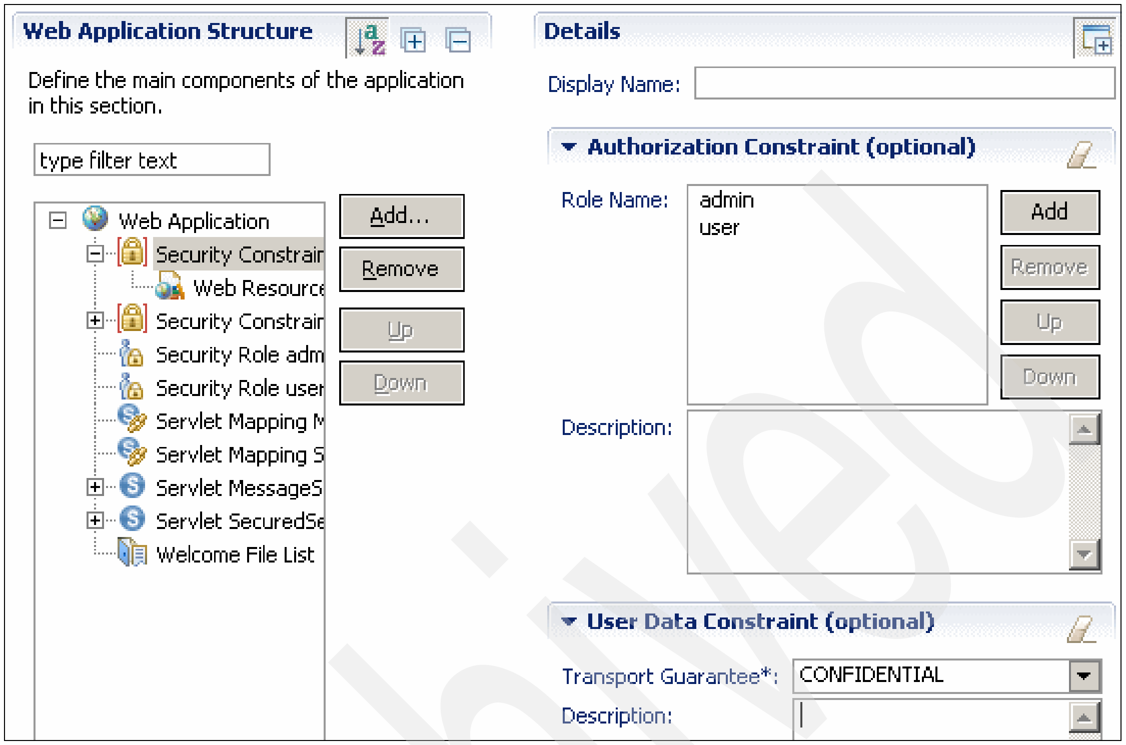


Figure 8-4 Editing Web resource collection element

5. Select **Security Constraint** and specify the roles that have access to the Web resources and the transport security requirements. In this example:

   – Role Name: `user` and `admin`
   – Description: `Constraint for user role.`
   – Transport Guarantee: `CONFIDENTIAL`

   Refer to Figure 8-5 on page 275.


Web Application Structure

Define the main components of the application in this section.

type filter text

- Web Application
  - Security Constrair
    - Web Resource
  - Security Constrair
  - Security Role adm
  - Security Role user
  - Servlet Mapping M
  - Servlet Mapping S
  - Servlet MessageS
  - Servlet SecuredSe
  - Welcome File List

Details

Display Name:

Authorization Constraint (optional)

Role Name: admin, user

Description:

User Data Constraint (optional)

Transport Guarantee*: CONFIDENTIAL

Description:

Figure 8-5 Editing Security Constraint element

6. It is a good practice to deny access to the application through other methods, such as DELETE, PUT, and so forth. Repeat steps 2-5 and add another security constraint with the following parameters:
   - HTTP Method list: `DELETE`, `PUT`, `TRACE`, and `OPTIONS`
   - In the URL Pattern list, add: `/*`
   - In the Web Resource Name, enter: `Denied resources`
   - Leave the Role Name list empty, but provide a description.
   - In the Authorization Constraint Description, enter: `Constraint for denied resources`. This action ensures that the authorization constraint is defined, but that no role is added.
   - In the Transport Guarantee, specify: `NONE`
7. Sometimes, it is required that several of the application files are available for unauthenticated users, for example, the start page, images, scripts, and so forth. You can make application files available for unauthenticated users by defining a security role that represents anonymous users (for example, an

`Everyone` role) and then specifying an additional security constraint with the following parameters:

– Add the HTTP Method list: `GET`, `POST`, and `HEAD`
– In the URL Pattern list, add: `/images/*`, `/index.jsp`, and so forth
– In the Web Resource Name, enter: `Anonymous resources`
– In the Role Name list, add the new role that represents anonymous users.

> **Tip:** The alternative to this process is to simply not define authorization constraints for these resources, which means that those resources can be accessed by unauthenticated users.
>
> However, it is more flexible to create a separate role that can be easily mapped during the application installation process to the `Everyone` or `All authenticated` special subjects based on the requirements.

– In the Transport Guarantee, specify `NONE` or `CONFIDENTIAL` depending on the transport restriction.

8. Save the file.

The `<security-constraint>` elements are added to the Web deployment descriptor as shown in Example 8-2. Select the **Source** tab in the editor to see the XML file.

Example 8-2 Security-constraint element in the deployment descriptor

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>all resources</web-resource-name>
      <url-pattern>/*</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
      <http-method>HEAD</http-method>
   </web-resource-collection>
   <auth-constraint>
      <role-name>user</role-name>
      <role-name>admin</role-name>
   </auth-constraint>
   <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
   </user-data-constraint>
</security-constraint>
```

Map users to security roles

After you define the security settings for the application, you have to map users from the user registry to the security roles; you can use Rational Application Developer or the application server during or after the application installation. For more information, refer to Chapter 7, "Application security" on page 253.

8.3 Programmatic security

This section talks about programmatic security, which means that the application is security-aware and contains calls to security APIs that allow the developer to make additional security decisions. Programmatic security is often used during Web application development, for example, to hide parts of the GUI that are not available to a particular role.

8.3.1 JEE security API

The javax.servlet.http.HttpServletRequest interface provides three methods that allow the Web developer to access security information about the user:

- String getRemoteUser() returns the user name that the client used to log in.
- java.security.Principal getUserPrincipal() returns the principal of the current user.
- Boolean isUserInRole(String roleNameReference) allows the developer to test if the calling user is in a specific role.

Using these methods: The getRemoteUser and getUserPrincipal methods return null for unauthenticated users. By default, they also return null for any unsecured resource, even for authenticated users, which however can be changed by the use of Web container settings. Refer to "Modifying Web authentication behavior" on page 322.

You can use the getRemoteUser and getUserPrincipal methods to get the name of the calling user as shown in Example 8-3 on page 278. The user name obtained can be displayed in the application user interface, or it can be stored in a log file or database to provide auditing of the user's actions in the application.

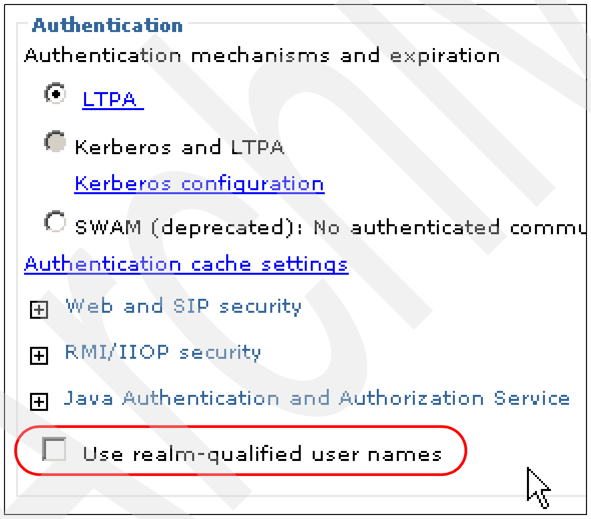
Example 8-3 Getting the user name in the JSP page

```
Remote user is: <%= request.getRemoteUser() %><BR>
<%
   String principal = null;
   if(request.getUserPrincipal() != null) {
      principal = request.getUserPrincipal().getName();

%>
User principal is: <%= principal %>
```

The user names that are obtained by both calls are usually the same, but not always. For example, if the application server is configured to return realm-qualified user names, getUserPrincipal will return the user name prefixed with the realm name. For example, defaultWIMFileBasedRealm/gas instead of gas.

This runtime setting is defined in the administrative console on the Global security page in the Authentication section as shown in Figure 8-6.


Authentication

Authentication mechanisms and expiration

[✓ LTPA]

Kerberos and LTPA

Kerberos configuration

SWAM (deprecated): No authenticated commu

Authentication cache settings

Web and SIP security

RMI/IIOP security

Java Authentication and Authorization Service

☐ Use realm-qualified user names

Figure 8-6 Enabling realm-qualified user names at run time

The isUserInRole method allows the developer to make access decisions in the method body based on the role of the current user. Example 8-4 on page 279 shows how to use this method to hide part of the user interface based on the user's role.

Example 8-4 Customizing interface based on user role

```
<% if(request.isUserInRole("admin")) { %>
     <p>This section is displayed if you have the admin role.</p>
<% } %>
<% if(request.isUserInRole("user")) { %>
     <p>This section is displayed if you have the user role.</p>
<% } %>
```

Security role reference: The parameter in the isUserInRole method is a security role reference. If the Web container does not find a role reference with this name, it will check the security role with the same name. Because of this algorithm and because only role names can be used in plain JSPs (only servlets and named JSPs, which are defined via the deployment descriptor, can have references), the security role reference usage is infrequent. You can still use security role references to increase the flexibility of changing role names in the application without having to recompile the servlet.

8.3.2 Defining security role references in the deployment descriptor

Security role references provide a level of indirection to isolate the role names that are used by the developers in the code and the actual runtime role names. Security role references are defined and linked to security roles using the Web deployment descriptor. A reference provides a way to bind a role name that is used in the code to a security role name in a descriptor.

The following steps describe how to create a security role reference and link it to security role:

1. Open the Web deployment descriptor and select the **Design** tab.
2. Servlets are automatically added to the Web deployment descriptor when you create them using the Create Servlet wizard. Expand **Web Application**, select the servlet to secure (for example, SecuredServlet), and click **Add**.
3. Select **Security Role Reference** as shown in Figure 8-7 on page 280, and click **OK**.

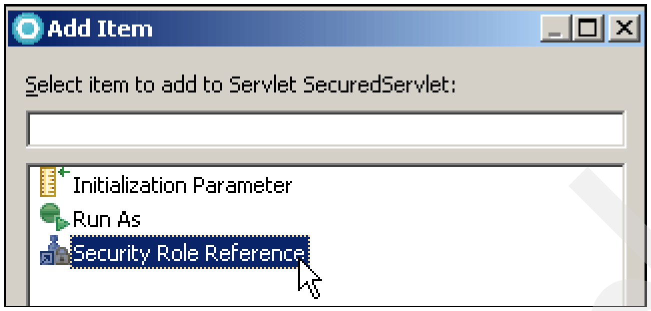
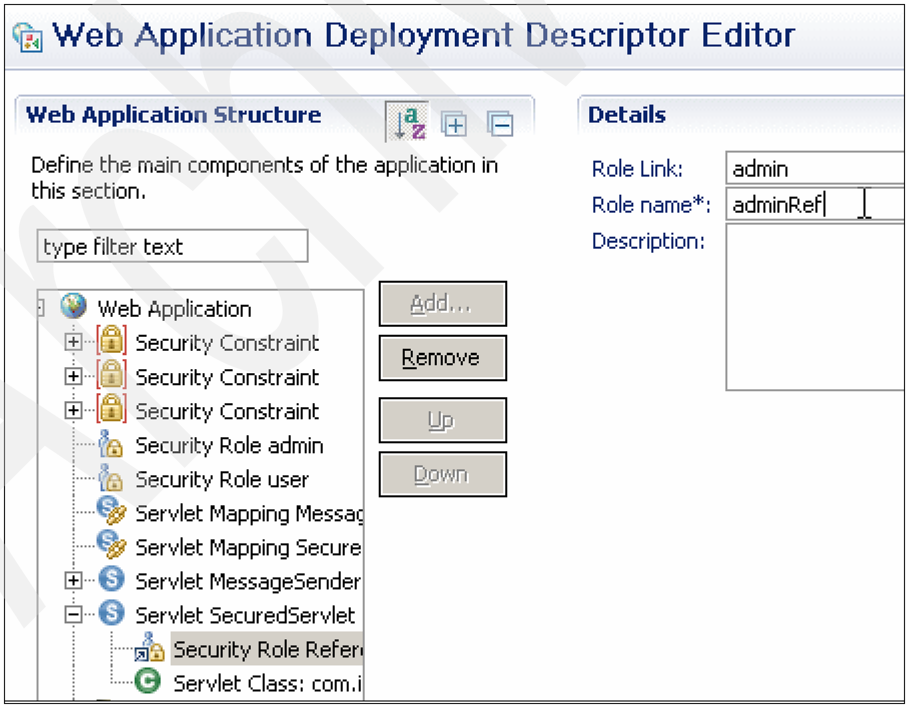


Figure 8-7 Adding Security Role Reference

4. Select the new security role reference element:
   - Enter the name of a defined security role in the role link field.
   - Enter a role name for the reference. This name will be used in the isUserInRole method, for example.

   In this example, `adminRef` is the role name and **`admin`** is the role link as shown in Figure 8-8.



Role Link: admin
Role name*: adminRef
Description:

Figure 8-8 Defining and linking security reference

5. Save the Web deployment descriptor.

A new `<security-role-ref>` element is added to the SecuredServlet definition in the Web deployment descriptor. It contains reference name and role name to which the reference is linked. Refer to Example 8-5.

Example 8-5 Security reference in the Web deployment descriptor

```
<servlet>
   <description></description>
   <display-name>SecuredServlet</display-name>
   <servlet-name>SecuredServlet</servlet-name>
   <servlet-class>com.ibm.itso.sample.security.servlet.SecuredServlet
   </servlet-class>
   <security-role-ref>
      <role-name>adminRef</role-name>
      <role-link>admin</role-link>
   </security-role-ref>
</servlet>
```

8.3.3 Defining security roles using annotations

Security roles can be also defined using the @DeclareRoles annotation. The @DeclareRoles annotation is specified at the class level and is used to define roles that are tested by calling isUserInRole.

Example 8-6 shows how to define a role reference using the @DeclareRoles annotation and then how to use it later in the Servlet method.

Example 8-6 DeclareRoles definition and usage sample

```
// Declaration of the manager role reference
@DeclareRoles({"manager"})
public class SecuredServlet extends HttpServlet {
   ....
   // usage of the declared role
   if(request.isUserInRole("manager")) {
         System.out.println("User is a member of 'manager' role <BR>");
         // .. do something extra for managers
   }
}
```

Any roles that are defined using the @DeclareRoles annotation must be mapped to users or groups from the user registry. This mapping can be done in Rational Application Developer or on the application server during or after the application installation. For more information, refer to Chapter 7, “Application security” on page 253.

8.4 Delegation

When a servlet calls a method in the EJB, the principal is, by default, propagated to the bean. Sometimes, for example, when the bean expects a specific identity, it is desirable to change the caller identity.

Delegation settings for the servlet can be defined using the @RunAs annotation or through the Web deployment descriptor. Delegation settings affect all calls from the servlet to the enterprise beans.

The RunAs role must be mapped to a real user at run time. Refer to Chapter 7, “Application security” on page 253 for more information about mapping users to roles.

8.4.1 Delegation using a deployment descriptor

Perform the following steps to define delegation using the Web deployment descriptor:

1. Open the Web deployment descriptor and select the **Design** tab.
2. Expand **Web Application**, select the servlet that will call the bean, and click **Add**.
3. Select the **Run As** element as shown in Figure 8-9, and click **OK**.

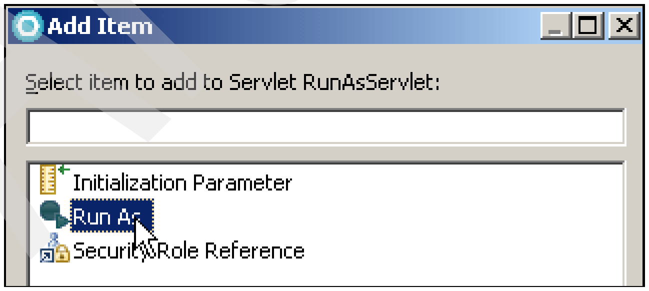


Figure 8-9 Adding Run As element

4. Select the new **Run As** element and type the role name (`admin` in this example) as shown in Figure 8-10.

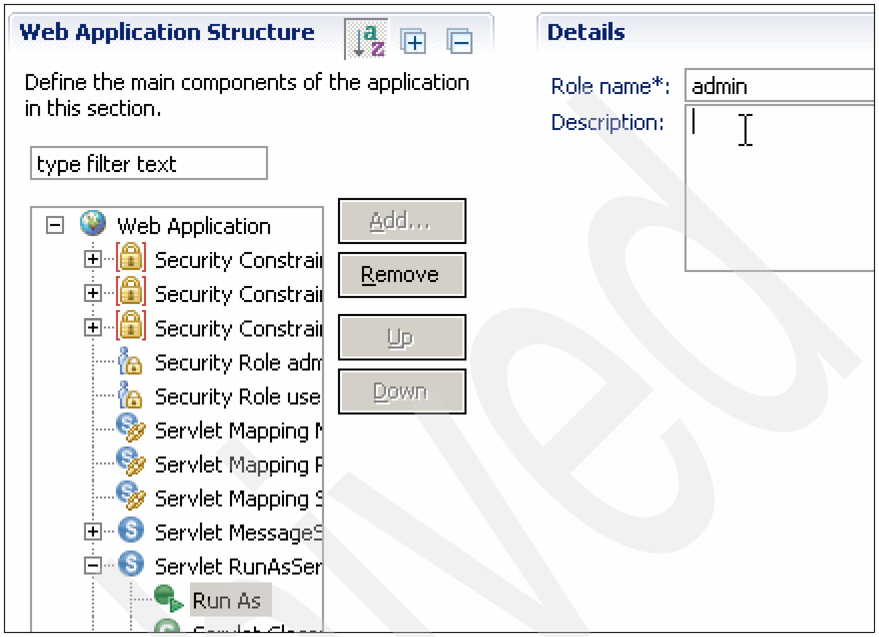


Figure 8-10 Specifying Run As role name

5. Save the Web deployment descriptor.

The new `<run-as>` element is added to the RunAsServlet definition in the Web deployment descriptor as shown in Example 8-7.

Example 8-7 Run As element in the Web deployment descriptor

```
<servlet>
   <description></description>
   <display-name>RunAsServlet</display-name>
   <servlet-name>RunAsServlet</servlet-name>
   <servlet-class>com.ibm.itso.sample.security.servlet.RunAsServlet
   </servlet-class>
   <run-as>
      <role-name>admin</role-name>
   </run-as>
</servlet>
```

8.4.2 Delegation using annotation

The @RunAs security annotation defines a role that will be used for delegation. This annotation can be specified in the servlet class as shown in Example 8-8. The role name that is given as the annotation value must have been defined using the DeclareRole annotation or configured as a security role in the Web deployment descriptor.

Example 8-8 Using @RunAs annotation in a servlet

```
import javax.annotation.security.DeclareRoles;
import javax.annotation.security.RunAs;

@RunAs("WebRunAsRole")
@DeclareRoles("WebRunAsRole")
public class RunAsServlet extends HttpServlet {
   ...
}
```

8.4.3 Annotation usage considerations

In JEE 5, only two annotations can be used in the Web module: @DeclareRoles and @RunAs. All other deployment information, such as servlet definitions, mappings, security constraints, login configuration, and so forth, must be specified in the Web deployment descriptor, which makes the use of annotations in Web modules questionable in terms of usefulness. However, they can be used in EJB modules, where in most cases they can replace the deployment descriptor completely.

8.5 Authentication mechanisms

The *authentication mechanism* defines how user credentials are sent to the application server. The JEE specification mentions the following mechanisms for Web authentication:

- HTTP basic authentication

  When this mechanism is used, the Web browser displays a built-in pop-up where the user enters a user name and password. Passwords are sent in the simple base64 encoding. The user authenticates with the target server, but the server does not authenticate with the user.

- Form-based authentication

  This mechanism allows a developer to provide a custom login page for the application. The user enters a user name and password, and then, the user submits the form. The user name and password combination is sent in plain text. The user authenticates with the target server, but the server does not authenticate with the user.

- Client certificate authentication

  This mechanism uses SSL certificates to authenticate both the server to the user and the user to the server. Communication is encrypted, and no user passwords are sent.

- HTTP digest authentication

  This mechanism is similar to HTTP basic authentication, but it sends passwords in the encrypted form. Its implementation is optional in the JEE specification, and WebSphere Application Server does not support it.

Security: You must always use HTTP basic and form-based authentication over a secured transport protocol, such as HTTPS, to provide the confidentiality of the user credentials.

Form-based authentication is preferred to HTTP basic, because HTTP basic caches credentials in the browser and the user cannot be logged out unless the browser is closed.

WebSphere additionally offers:

- SPNEGO Web authentication

  This mechanism uses Kerberos tokens and Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) to authenticate users. SPNEGO is most often used to enable desktop single sign-on (SSO), where the user does not have to authenticate to the Web application after the user is authenticated in the Active Directory® domain. The user’s passwords are not sent, only the tokens.

- Trust association

  This mechanism allows the use of an external authentication proxy, for example, IBM Tivoli WebSEAL, or the use of custom tokens to authenticate users. Refer to “Trust association interceptor” on page 316.

- JAAS authentication

  WebSphere Application Server provides plug-in points to insert custom login modules that allow customized authentication. Refer to 8.8, "Customizing the login process" on page 315.

So far in our example, we have not provided any authentication mechanism settings, yet we can log in and use the application, which is the result of the default authentication mechanism, which is HTTP basic authentication. If you define at least one security constraint and no authentication mechanism, the default is used.

For any type of authentication to work, at least one security constraint for the requested Web resource must be defined and application security must be enabled on the application server.

8.6 Configuring form-based authentication

With form-based authentication, the following actions occur when the user requests a protected resource:

1. The application server checks to see if user is authenticated.
2. If the user is unauthenticated, the server redirects to the login page. If the user is authenticated, the resource is shown and the rest of this process is skipped.
3. The user provides the login credentials and submits the form.
4. The server performs authentication.
5. If authentication is successful, the user is redirected to the originally requested resource. Otherwise, the error page is displayed.

SSO: Form-based authentication will fail if single sign-on (SSO) is not enabled in global security. SSO is enabled by default. To check the current setting from the administrative console, go to **Global Security** → **Web and SIP security** → **Single sign-on (SSO)**. Ensure that the Enabled check box is selected.

Perform the following steps to configure form-based authentication:

1. Open the Web deployment descriptor.
2. Select **Web Application** and click **Add**.
3. Select **Login Configuration** as shown in Figure 8-11 on page 287, and click **OK**.

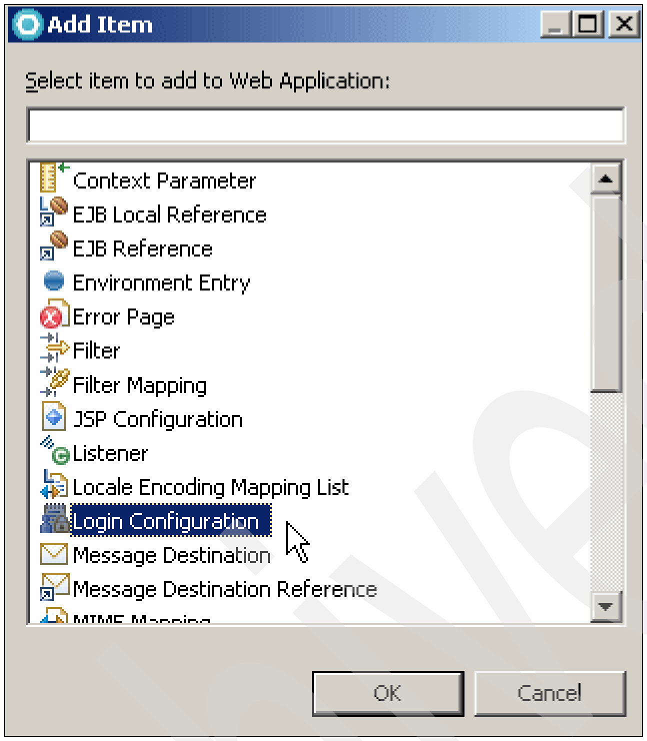
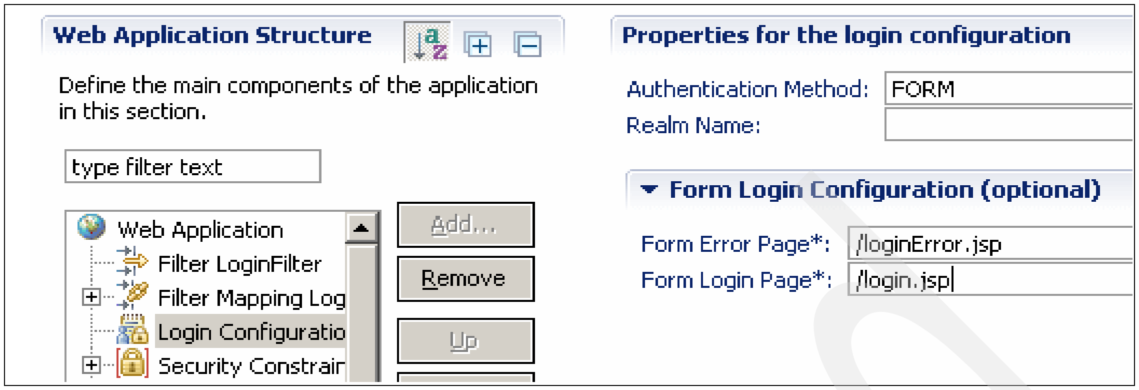


Figure 8-11   Adding Login Configuration element

4. Select the **Login Configuration** and specify one of the following combination of values to select the type of authentication to use:
   - For HTTP basic authentication: Enter `BASIC` as the authentication method and *yourRealmName* as the realm name. The realm name will be displayed in the authentication pop-up window that is shown by the browser.
   - For client certificate authentication: Enter `CLIENT-CERT` as the authentication method. Refer to 8.7, “Configuring client certificate authentication” on page 294 for more details.
   - For form-based authentication: Enter `FORM` as the authentication method, and enter the JSPs to use for login and login errors.

   For this example, we configured form-based authentication as shown in Figure 8-12 on page 288. We used `/login.jsp` as the login page and `/loginError.jsp` as the login error page.


Web Application Structure
Define the main components of the application in this section.
type filter text
Web Application
Filter LoginFilter
Filter Mapping Log
Login Configuratio
Security Constrair
Add...
Remove
Up
Properties for the login configuration
Form Login Configuration (optional)

Authentication Method: FORM
Realm Name:
Form Error Page*: /loginError.jsp
Form Login Page*: /login.jsp

Figure 8-12 Configuring authentication mechanism

5. Save the Web deployment descriptor.

The new `<login-config>` element is added to the Web deployment descriptor as shown in Example 8-9.

Example 8-9 Login config element in the Web deployment descriptor

```
<login-config>
   <auth-method>FORM</auth-method>
   <form-login-config>
      <form-login-page>/login.jsp</form-login-page>
      <form-error-page>/loginError.jsp</form-error-page>
   </form-login-config>
</login-config>
```

8.6.1 Building the login page

The login and error pages can be JSPs or HTML pages. The JEE specification determines how the login form needs to be built:

- The form must be sent via a post method.
- The form's action must be `j_security_check`.
- The user name must be sent via the `j_username` parameter.
- The password must be sent via the `j_password` parameter.

A sample login form is shown in Example 8-10 on page 289.

Example 8-10   Sample login form

```
...
<form action="j_security_check" method="post">
Username: <input type="text" name="j_username" size="20"><br>
Password: <input type="password" name="j_password" size="20"><br>
   <input type="submit" value="Login">
</form>
...
```

Figure 8-13 shows how this form looks in the browser.

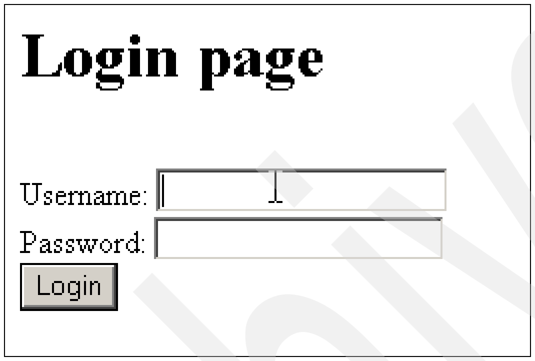


Figure 8-13   Sample login page displayed in the browser

8.6.2 Getting the login exception details

An error page usually displays general information about a login error and provides the ability to retry the login. It does not provide a method to get information about the cause of the exception, because it is displayed using redirect.

To get the root cause of the exception, use the WSSubject.getRootLoginException()method in combination with a servlet filter.

Note: Providing the login failure reason to the user is a security exposure and is not recommended. This technique is more appropriate for problem determination situations.

You can also code your application to display general information, for example, an error number, without revealing the stack trace or exact reason. The error number can be used to contact application support without providing the server log.

The WSSubject.getRootLoginException() returns the exception caught during system login. This exception can contain several nested exceptions. To extract the real root cause, use the method that is shown in Example 8-11.

Example 8-11   Getting the root cause of the login exception

```
import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.auth.WSLoginFailedException;
import javax.naming.NamingException;
...
public Throwable determineCause(Throwable e) {
   Throwable rootEx = e, tempEx = null;
   // keep looping until there are no more embedded
   // WSLoginFailedException or WSSecurityException exceptions
   while (true) {
      if (e instanceof WSLoginFailedException) {
         tempEx = ((WSLoginFailedException)e).getCause();
      }
      else if (e instanceof WSSecurityException) {
         tempEx = ((WSSecurityException)e).getCause();
      }
      else if (e instanceof NamingException) {
         // check for Ldap embedded exception
         tempEx = ((NamingException)e).getRootCause();
      }
      else {
         // this is the root from the WebSphere
         //  Application Server perspective
         return rootEx;
      }
      if (tempEx != null) {
         // we have nested exception, check it
         rootEx = tempEx;
         e = tempEx;
         continue;
      }
      else {
         // the cause was null, return parent
         return rootEx;
      }
   }  //while
}
```

We have the cause, so now we need to find a way to propagate it to an error page. We will use HttpServletResponseWrapper, a cookie, and the servlet filter. We discuss servlet filters in “Login filter” on page 315.

First, we need to use HttpServletResponseWrapper to overcome committing the response by the j_security_check servlet. It is a simple wrapper that overrides the sendRedirect method to store the location and defines its own method for sending a redirect. It is shown in Example 8-12. This class can be a inner class in your login filter.

Example 8-12 HttpServletResponseWrapper

```
class MyWrapper extends HttpServletResponseWrapper {
   String originalRedirect;
   public MyWrapper(HttpServletResponse response) {
      super(response);
   }
   @Override
   public void sendRedirect(String location) throws IOException {
      // just store location, don’t send redirect to avoid
      // committing response
      originalRedirect = location;
   }
   // use this method to send redirect after modifying response
   public void sendMyRedirect() throws IOException {
      super.sendRedirect(originalRedirect);
   }
}
```

Now, use the wrapper in the servlet filter to set the cookie as shown in Example 8-13.

Example 8-13 Passing exception message by using a cookie

```
public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
   //create wrapper
   MyWrapper myRes = new MyWrapper((HttpServletResponse) response);
   // call authentication
   chain.doFilter(request, myRes);
   // check for login error
   Throwable t = WSSubject.getRootLoginException();
   if (t != null) {
      t = determineCause(t);
      Cookie c = new Cookie("loginError", t.getMessage());
      c.setMaxAge(-1);
      myRes.addCookie(c);
   }
   else {
      // authentication successful, remove the cookie
      Cookie c = new Cookie("loginError", "");
      c.setMaxAge(0);
      myRes.addCookie(c);
   }
   // now it is safe to send redirect
   myRes.sendMyRedirect();
}
```

The last task is to retrieve the cookie and display the message in the error page, as shown in Example 8-14.

Example 8-14 Displaying the exception in the error page

```
...
<% Cookie cookies[] = request.getCookies();
for(int i = 0; i < cookies.length; i++) {
   Cookie c = cookies[i];
   if(c.getName().equals("loginError")) {
      out.println(c.getValue());
   }
}
%>
...
```

Using a session: Similar functionality can be achieved using a session instead of a cookie. The downside of the session solution is that it generates a session for every authentication request, successful or not, and it can be used by a malicious attacker to generate a fake authentication request to overwhelm the server with sessions.

8.6.3 Logout

Almost every modern Web application requires logout functionality to ensure that the users' private data is securely cleared. WebSphere Application Server has an extension to the JEE specification that allows you to log out an authenticated user. You can use the form logout or the programmatic logout.

Logout: Logout only works with the form-based login. When the application is configured to use basic authentication, the credentials are stored in the client's browser and the browser sends the user name and password to the server together with every request. The only way to log out is to break the session by closing the browser.

Form logout

Similarly to the login page, the logout page must contain the form with specific parameters:

- The form must be sent via the POST method.
- The form's action must be `ibm_security_logout`.
- The optional hidden parameter `logoutExitPage` can be used to define a page that is displayed after the logout. This parameter permits a relative or fully qualified URL. If no exit page is specified, a default HTML logout message is returned.

A sample logout form is shown in Example 8-15.

Example 8-15 Logout form

```
<FORM METHOD=POST ACTION="ibm_security_logout" NAME="logout">
   <input type="submit" name="logout" value="Logout">
   <input type="HIDDEN" name="logoutExitPage" value="/index.jsp">
</form>
```

When the form is posted, the application server performs the following actions:

- Clears the Lightweight Third Party Authentication (LTPA)/single sign-on (SSO) cookies
- Invalidates the HTTP session
- If an exit page is defined, redirects the user to the page

Programmatic logout

If you want to combine a logout with an action in the application, you can use the WebSphere security API to log out the user. The WSSecurityHelper.revokeSSOCookies() method removes SSO cookies.

Example 8-16 on page 294 shows a sample logout method.

Example 8-16 Programmatic logout

```
public void logout(HttpServletRequest requset, HttpServletResponse
response)  throws ServletException {
   // invalidate session
   if(request.getSession(false) != null) {
      request.getSession(false).invalidate();
   }
   // remove SSO cookies
   WSSecurityHelper.revokeSSOCookies(request, response);
}
```

Session considerations: In programmatic logout, invalidating the session is critical. If the session is not invalidated and the user does not exit the browser, the next logged in user will be able to see data from the previous session.

Refer to 8.9.2, “Session security integration” on page 320 for information about additional session authentication that can be enabled.

8.7 Configuring client certificate authentication

Client certificate authentication sets up a mutual trust between the user and the server. Each party identifies with the certificate, which must be trusted or signed by a trusted certificate authority (CA).

The most common scenario is to use a Web server in the DMZ as a Secure Sockets Layer (SSL) terminator. The Web server is responsible for client authentication and certificate validation. Mutual trust is establish between the user’s browser and the Web server. The WebSphere Application Server plug-in installed with the Web server is responsible for passing authentication data to the application server. For security reasons, we recommend that you establish mutual SSL between the plug-in and the Web container also.

To successfully execute client certificate authentication, you must configure the following components:

- Application
- Application server
- Web server
- Browser

8.7.1 Application configuration

To configure the client certificate authentication mechanism, repeat steps 1- 4 from 8.5, “Authentication mechanisms” on page 284. In step 4, specify `CLIENT-CERT` as the authentication method.

The `<login-config>` element from the Web deployment descriptor for certificate authentication is shown in Example 8-17.

Example 8-17 Login configuration for client certificate authentication

```
<login-config>
   <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

8.7.2 Application server configuration

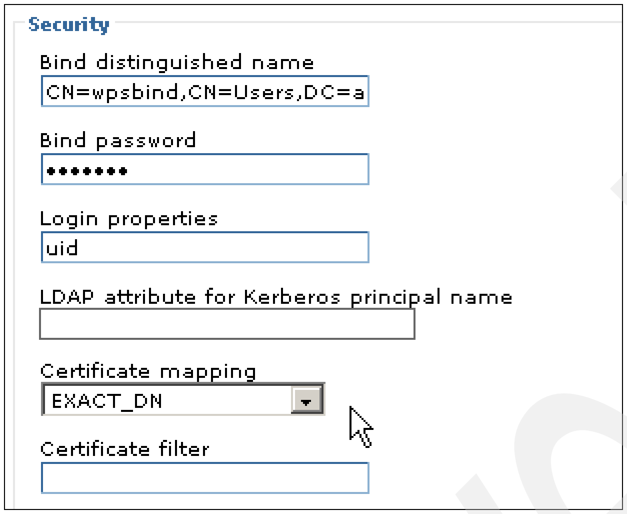
After the application is configured to use client certificate authentication, the application server must be able to support the client certificate authentication.

User registry

First, you have to make sure that the application server uses a user registry that supports client certificate authentication. Currently, the local operating system and built-in file-based registries do not support it. If you want to use federated repositories, you have to remove the default file repository and add Lightweight Directory Access Protocol (LDAP) registries. After the registry is configured, the certificate mapping options must be set. These options determine how the client certificate is mapped to the LDAP registry.

To set these options from the administrative console, navigate to the configuration page for global security or, if domains are used, for the security domain to be used for the application server. Select the user registry type in the realm type field and click **Configure**.

For an LDAP registry in a federated repository, the mapping options can be found in the main repository page in the Security section as shown in Figure 8-14 on page 296.



Security

Bind distinguished name
CN=wpsbind,CN=Users,DC=a

Bind password
•••••••

Login properties
uid

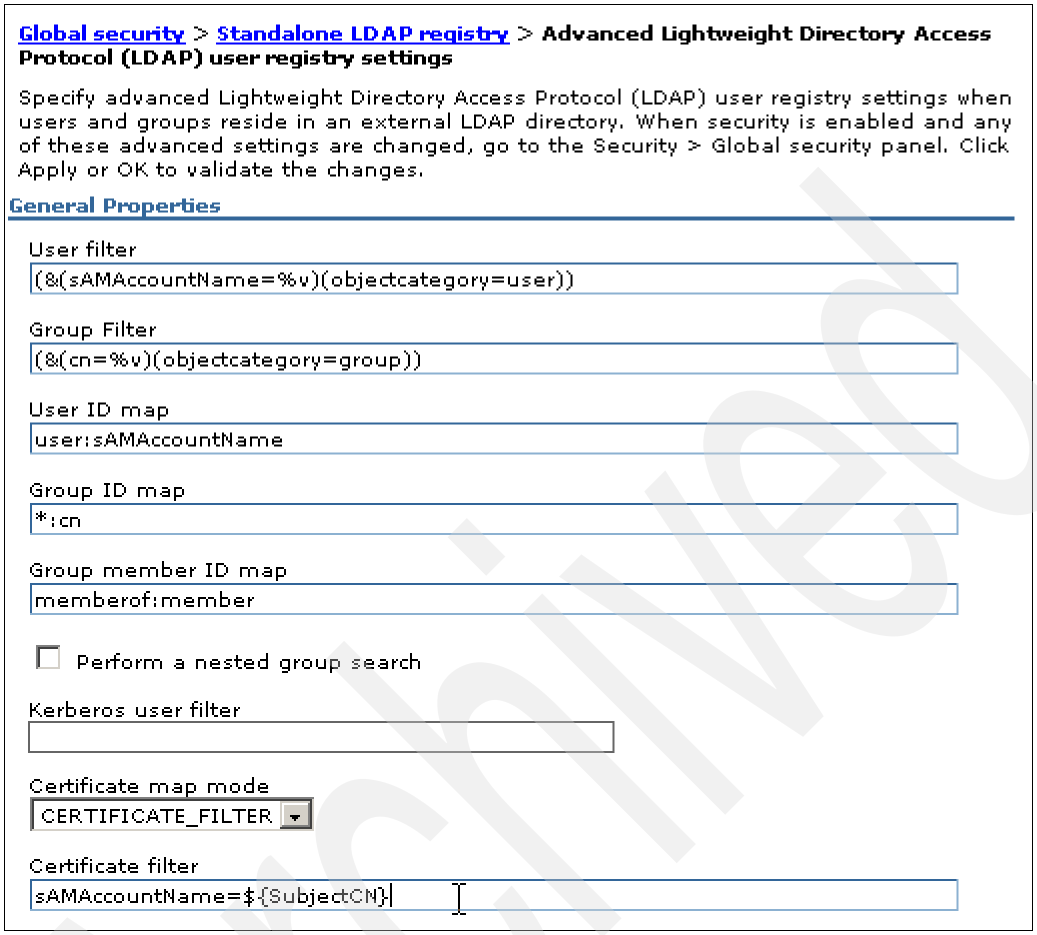
LDAP attribute for Kerberos principal name

Certificate mapping
EXACT_DN

Certificate filter

Figure 8-14 Certificate mapping settings in the LDAP federated repository

For a stand-alone LDAP registry, the certificate mapping settings are in the Advanced LDAP user registry settings as shown in Figure 8-15 on page 297.


Global security > Standalone LDAP registry > Advanced Lightweight Directory Access Protocol (LDAP) user registry settings

Specify advanced Lightweight Directory Access Protocol (LDAP) user registry settings when users and groups reside in an external LDAP directory. When security is enabled and any of these advanced settings are changed, go to the Security > Global security panel. Click Apply or OK to validate the changes.

General Properties

User filter
(&(sAMAccountName=%v)(objectcategory=user))

Group Filter
(&(cn=%v)(objectcategory=group))

User ID map
user:sAMAccountName

Group ID map
*:cn

Group member ID map
memberof:member

☐ Perform a nested group search

Kerberos user filter

Certificate map mode
CERTIFICATE_FILTER

Certificate filter
sAMAccountName=${SubjectCN}

Figure 8-15 Certificate mapping settings in the Standalone LDAP registry

There are two options for certificate mapping:

- EXACT_DN

  If EXACT_DN is selected, the distinguished name (DN) in the certificate must exactly match the user entry in the LDAP server, including case and spaces. For example, if the user DN is cn=john,ou=users,o=ITSO,c=US, John has to be under the users organizational unit (ou), ITSO organization (o), US country (c).

  It is extremely hard or even impossible to support certificates that are already owned by users and that were issued by trusted CAs. In that case, you need to use the next option.

- CERTIFICATE_FILTER

  If `CERTIFICATE_FILTER` is selected, specify the LDAP filter for mapping attributes in the client certificate to entries in LDAP. The filter must match one LDAP entry at run time or authentication fails.

  The syntax of the filter is:

  `LDAP attribute``=${`*`Client certificate attribute`*`}`

  The *`LDAP attribute`* portion of the filter specification is an LDAP attribute that depends on the LDAP server's schema.

  The *`Client certificate attribute`* side of the filter specification is one of the public attributes in the client certificate. The Client certificate attribute must begin with a dollar sign (`$`), open bracket (`{`), and end with a close bracket (`}`), for example, `uid=${SubjectCN}`.

  The following client certificate attributes can be used:

  – `${UniqueKey}`
  – `${PublicKey}`
  – `${IssuerDN}`
  – `${Issuer`*`xx`*`}` - where *xx* is replaced by the characters that represent any valid component of the issuer distinguished name. For example, use `${IssuerCN}` for the issuer common name.
  – `${NotAfter}`
  – `${NotBefore}`
  – `${SerialNumber}`
  – `${SigAlgName}`
  – `${SigAlgOID}`
  – `${SigAlgParams}`
  – `${Version}`
  – `${SubjectDN}`
  – `${Subject<`*`xx`*`>}` - where <*xx*> is replaced by the characters that represent any valid component of the Subject Distinguished Name. For example, use `${SubjectCN}` for the Subject Common Name.

  For example, assume:

  – The user's certificate DN is:

    `cn=wpsbind, o=test`

  – The user's LDAP DN is:

    `CN=wpsbind,CN=Users,DC=ad,DC=test,DC=com,DC=pl`

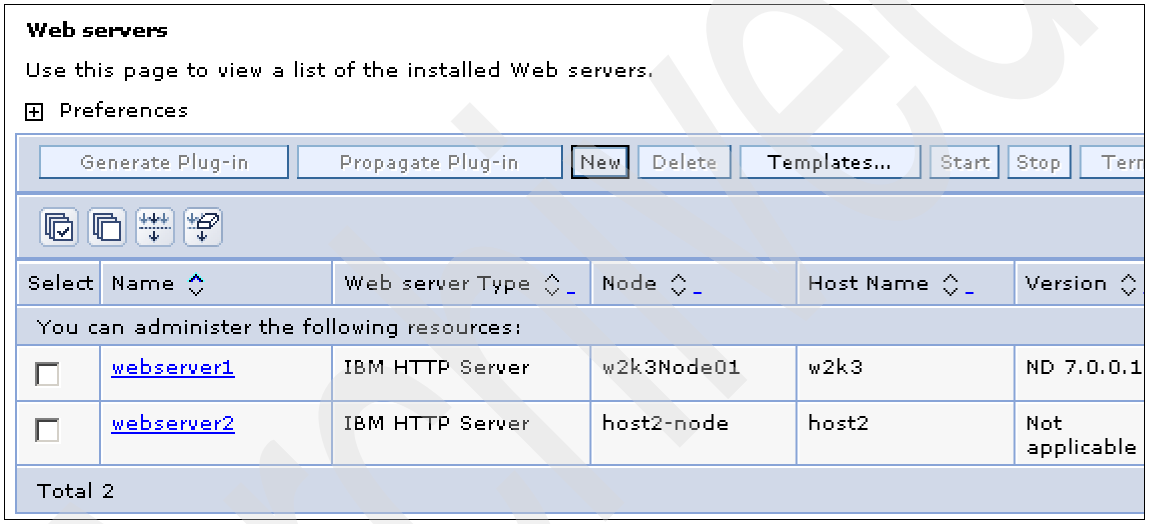
The filter can be `uid=${SubjectCN}`.

Web server plug-in configuration

This is an optional step. You can manage the Web server directly through its configuration files, but we recommend that you use the administrative console, because it is more efficient and elegant.

This process assumes that you have an IBM HTTP Server with the Web server plug-in installed and that the Web server is defined in the WebSphere administrative console. To configure the Web server plug-in:

1. Click **Servers** → **Server Types** → **Web servers** to see the Web servers list as shown in Figure 8-16.



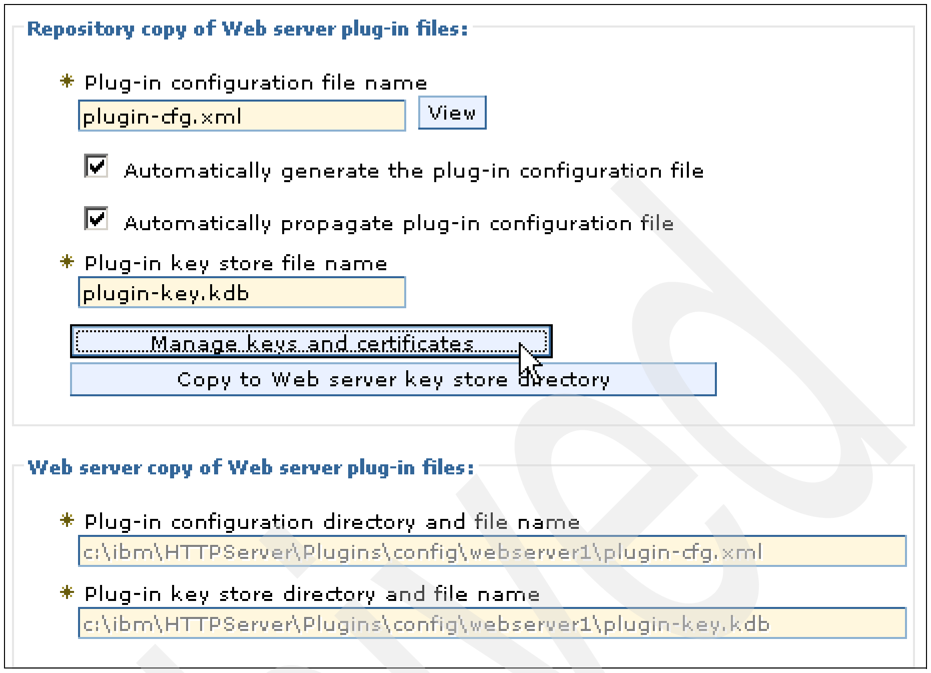
Select	Name	Web server Type	Node	Host Name	Version
You can administer the following resources:					
☐	webserver1	IBM HTTP Server	w2k3Node01	w2k3	ND 7.0.0.1
☐	webserver2	IBM HTTP Server	host2-node	host2	Not applicable
Total 2					

Figure 8-16 Web servers list

2. Click the Web server name to open the configuration page.
3. Verify the signers are in the plug-in keystore.

   The Web server plug-in must be able to connect to server with the application (servers in the case of the cluster), so it must have the server certificate in its truststore. In the case of a default SSL configuration for the nodes, it is enough to add the root certificate to the truststore, because all certificates have it in the chain.

   Click **Servers** → **Server Types** → **Web servers** → ***webserverName*** → **Plug-in properties** to open the plug-in properties page, and click **Manage keys and certificates** as shown in Figure 8-17 on page 300.


Repository copy of Web server plug-in files:

* Plug-in configuration file name: plugin-cfg.xml

☑ Automatically generate the plug-in configuration file

☑ Automatically propagate plug-in configuration file

* Plug-in key store file name: plugin-key.kdb

Manage keys and certificates
Copy to Web server key store directory

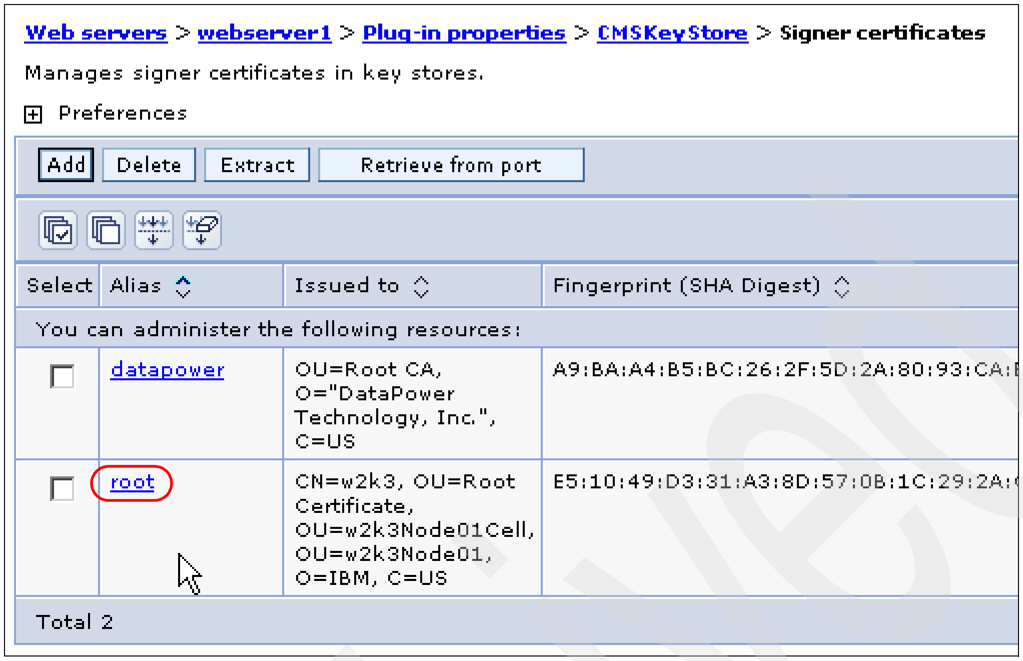
Web server copy of Web server plug-in files:

* Plug-in configuration directory and file name: c:\ibm\HTTPServer\Plugins\config\webserver1\plugin-cfg.xml

* Plug-in key store directory and file name: c:\ibm\HTTPServer\Plugins\config\webserver1\plugin-key.kdb

Figure 8-17 Plug-in properties

4. Click **Signer certificates** and look for **root** as shown in Figure 8-18 on page 301.


Web servers > webserver1 > Plug-in properties > CMSKeyStore > Signer certificates

Manages signer certificates in key stores.

⊞ Preferences

Select	Alias	Issued to	Fingerprint (SHA Digest)
You can administer the following resources:			
☐	datapower	OU=Root CA, O="DataPower Technology, Inc.", C=US	A9:BA:A4:B5:BC:26:2F:5D:2A:80:93:CA:
☐	root	CN=w2k3, OU=Root Certificate, OU=w2k3Node01Cell, OU=w2k3Node01, O=IBM, C=US	E5:10:49:D3:31:A3:8D:57:0B:1C:29:2A:
Total 2			

Figure 8-18 Checking root signer certificate

You can safely remove any other certificates; they are not needed. They are added by default when the keystore is created.

If there is no root certificate or custom certificates have been created for the servers, the certificate has to be added to the truststore by a signers exchange between the server keystore (NodeDefaultKeyStore) and the plug-in truststore (CMSKeyStore). The default store names are given in parentheses. Figure 8-20 on page 306 shows how to exchange signers.

5. Create the certificate for the plug-in (optional).

   The plug-in needs its own certificate to authenticate to the server. When you create the Web server definition, a CMSKeystore is created that contains a chained certificate for the plug-in. It is signed by the default root, so the cell will trust it, and because it is chained, it can be easily replaced when it expires. This step is only required if you do not want the certificate signed by the default root, for example, in isolation scenarios.

   You can create a self-signed certificate for this purpose:

   a. Click **CMSKeyStore** in the navigation trail to display the plug-in’s keystore details.

   b. Click **Personal certificates**. A page with a list of personal certificates is opened.

   c. Click **Create** and select **Self-signed Certificate**.
   d. Specify the certificate details, and click **OK**.

6. Set the default certificate.

   To successfully authenticate, the plug-in has to send its personal certificate to the application server. The plug-in will only send the certificate marked as the default. If there is no default certificate, authentication will fail. Currently, there is no way to set the default certificate by using the the administrative console, and the ikeyman tool must be used. This step must be done regardless of whether you use the certificate created for you or create your own certificate.

   Perform the following steps to configure the default certificate for the plug-in:

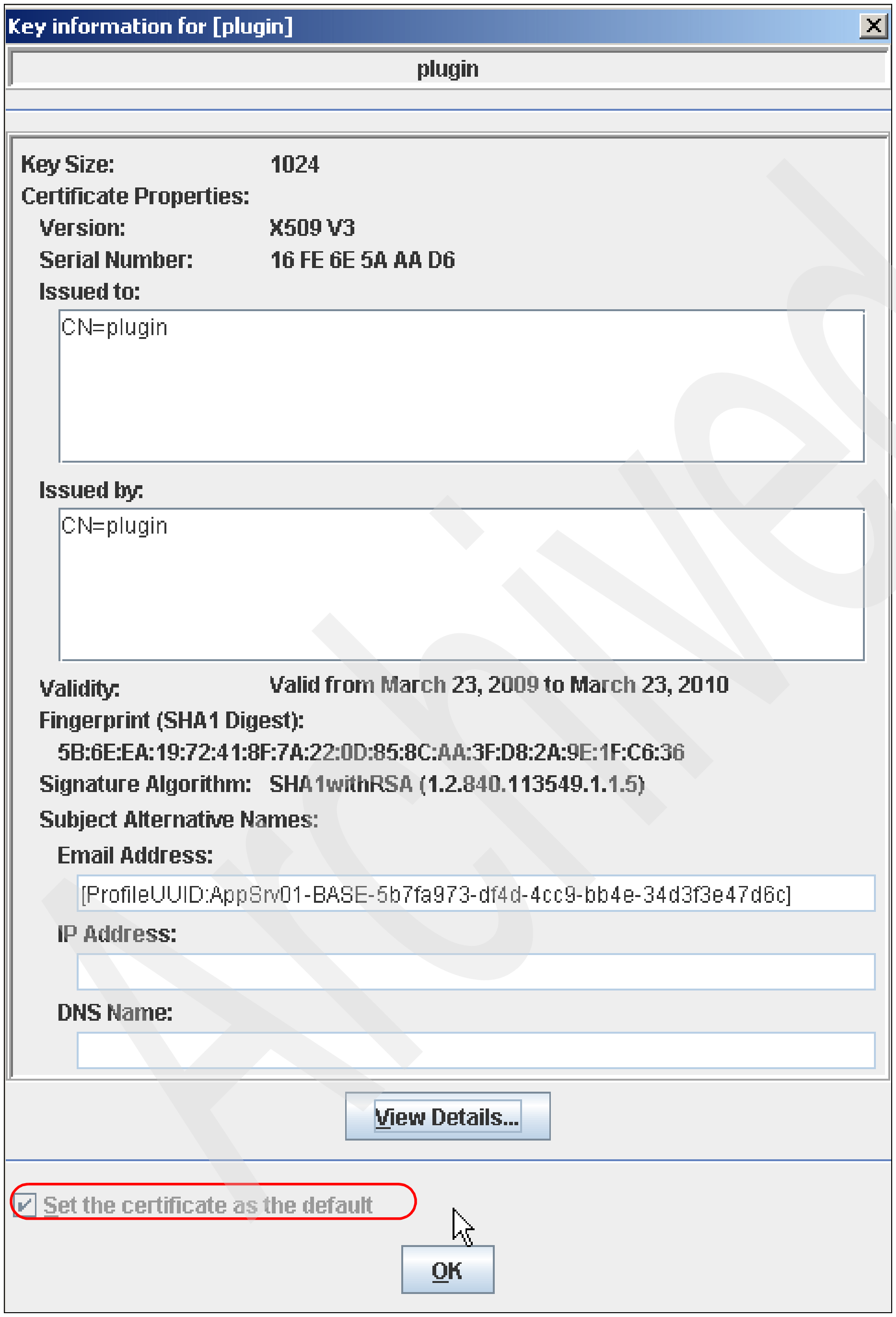
   a. Start ikeyman using the following command:

      ```
      was_install_root\bin\ikeyman.bat
      ```

   b. Open the plug-in keystore file that is located in:

      ```
      profile_root\config\cells\cellName\nodes\nodeName\servers\webserv
      er\plugin-key.kdb
      ```

   c. Enter the keystore password and click **OK** (the default password is `WebAS`).
   d. Select **Personal Certificates** from the drop-down list.
   e. Double-click the self-signed certificate, check **Set the certificate as the default** as shown in Figure 8-19 on page 303, and click **OK**.


Key information for [plugin]

plugin

Key Size: 1024
Certificate Properties:
Version: X509 V3
Serial Number: 16 FE 6E 5A AA D6
Issued to:
CN=plugin
Issued by:
CN=plugin
Validity: Valid from March 23, 2009 to March 23, 2010
Fingerprint (SHA1 Digest):
5B:6E:EA:19:72:41:8F:7A:22:0D:85:8C:AA:3F:D8:2A:9E:1F:C6:36
Signature Algorithm: SHA1withRSA (1.2.840.113549.1.1.5)
Subject Alternative Names:
Email Address:
[ProfileUUID:AppSrv01-BASE-5b7fa973-df4d-4cc9-bb4e-34d3f3e47d6c]
IP Address:
DNS Name:

[✓] Set the certificate as the default

Figure 8-19 Setting the default certificate

   f. Close the keystore file.

7. Transfer the plug-in files.

   When edited by using the administrative console, the plug-in configuration file and the keystore files are stored in the application server's config directory. These files need to be transferred to the appropriate directory on the Web server machine, where the plug-in itself is installed.

   Follow these steps to transfer the files for managed Web servers using the administrative console:

   – Copy the keystore files:

      i. Select **Servers** → **Server Types** → **Web servers** → ***webserverName*** → **Plug-in properties** to open the plug-in properties page.

      ii. Click **Copy to Web server keystore directory** on the plug-in properties page. Refer to Figure 8-17 on page 300.

   – Copy the plug-in configuration file.

      Perform these steps only if automatic generation and propagation are disabled on the plug-in properties page:

      i. Click **Servers** → **Server Types** → **Web servers**.

      ii. Check the Web server name on the Web server list (refer to Figure 8-16 on page 299), and click **Generate Plug-in**.

      iii. Then, check the Web server name and click **Propagate Plug-in**.

   – For unmanaged Web servers, copy the `plugin-cfg.xml`, `plugin-key.kdb`, and `plugin-key.sth` files from:

      profile_root`\config\cells\`*cellName*`\nodes\`*nodeName*`\servers\`*webserver*`\`

      to the following folder on the Web server machine:

      plugin_root`\config\`*webserver*

> **Recommendation:** For DMZ plug-in deployments, we recommend that you copy the files manually by using **`ssh`** or **`scp`** and disable the HTTP administration service.

Web container configuration

The plug-in sends sensitive user data to the Web container. This channel must be secured. We recommend that you configure a mutual SSL between the plug-in and the Web container. With this configuration, the Web container accepts connections only from trusted Web servers.

For this configuration, you have to create a new truststore and SSL configuration. The SSL configuration must require client authentication. In this chapter, we only provide the general configuration steps. Refer to Chapter 5, “Secure Sockets Layer administration” on page 151 for details about how to manage SSL configurations.

Follow these steps to perform the required changes through the administrative console:

1. Create a new truststore.

   A new PKCS12 truststore can be created on the following page in the administrative console by selecting:

   Security → SSL certificate and key management → Manage endpoint security configurations → Inbound → ***managementScope*** **→ keystores and certificates**

   This new truststore will be used only for the client certificate authentication. The recommended management scope depends on how the truststore will be used:

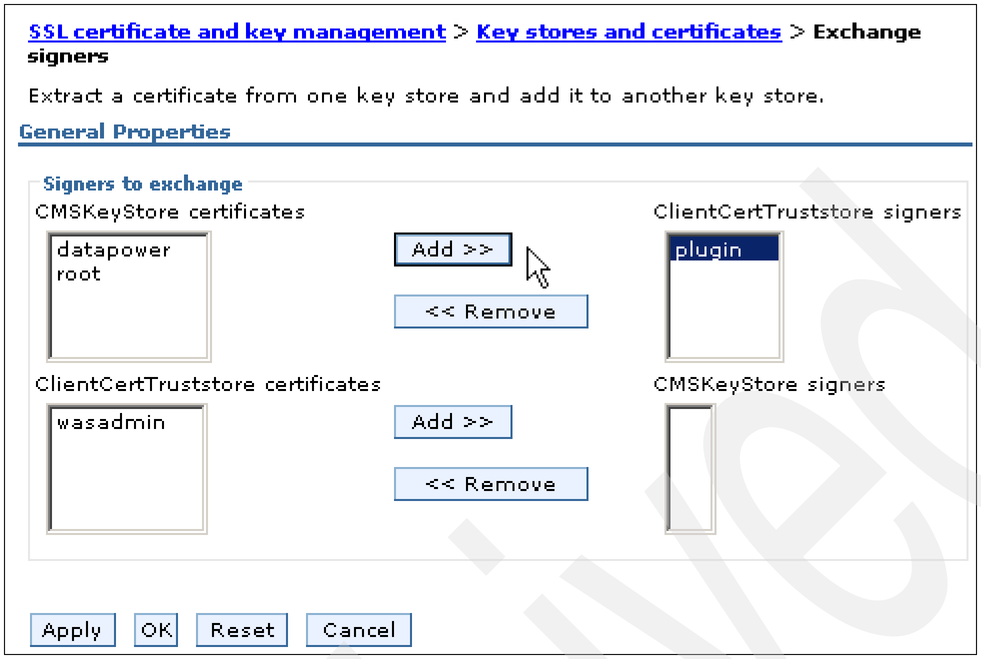
   – *Cell* for clustered servers
   – *Node* for many servers on selected node
   – *Server* for specific server

2. Add the signer certificates to the truststore.

   The Web container must only accept connections from trusted Web servers. Therefore, add each plug-in certificate as a signer in the Web container truststore:

   a. Click **Security → SSL certificate and key management → keystores and certificates**.
   b. Select the plug-in keystore (CMSKeyStore) and the created truststore, and then, click **Exchange Signers**.
   c. Select the plug-in certificate from CMSKeyStore and click **Add** to add it to the truststore as a signer certificate. If you use the default generated plug-in certificate, which is chained, add the signer instead of the plug-in certificate.

      Refer to Figure 8-20 on page 306.


SSL certificate and key management > Key stores and certificates > Exchange signers

Extract a certificate from one key store and add it to another key store.

General Properties

Signers to exchange

CMSKeyStore certificates: datapower, root

ClientCertTruststore signers: plugin

ClientCertTruststore certificates: wasadmin

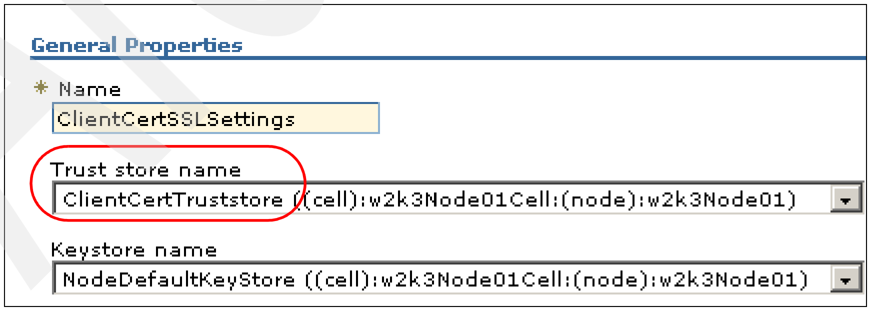
CMSKeyStore signers:

Figure 8-20 Exchanging signers

3. Create a new SSL configuration at the same scope as the truststore:

   a. Go to **Security** → **SSL certificate and key management** → **Manage endpoint security configurations** → **Inbound** → ***managementScope*** → **SSL configurations**, and click **New**.

   b. Associate the configuration with the new truststore. For the keystore, you can leave the default setting as shown in Figure 8-21.


General Properties

* Name: ClientCertSSLSettings

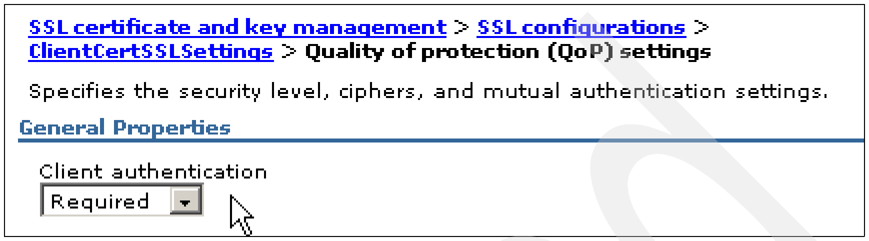
Trust store name: ClientCertTruststore ((cell):w2k3Node01Cell:(node):w2k3Node01)

Keystore name: NodeDefaultKeyStore ((cell):w2k3Node01Cell:(node):w2k3Node01)

Figure 8-21 Creating a new SSL configuration

4. Set the client authentication.

   Configure this option by selecting **Security** → **SSL certificate and key management** → **SSL configurations** → ***SSLSettingsName*** → **Quality of protection (QoP) settings** as shown in Figure 8-22.



SSL certificate and key management > SSL configurations > ClientCertSSLSettings > Quality of protection (QoP) settings

Specifies the security level, ciphers, and mutual authentication settings.

General Properties

Client authentication
Required

Figure 8-22 Selecting client authentication

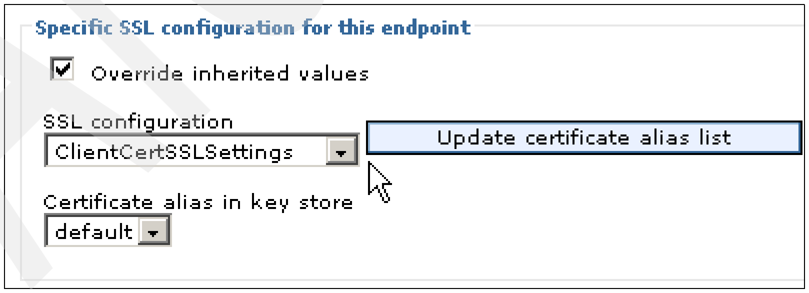
   In Figure 8-22, select **Required** for Client authentication. With this option, the server requires the client to send a certificate; otherwise, the connection fails. You must select this option to establish mutual trust between the plug-in and the Web container and to ensure that only trusted plug-ins can connect.

5. Apply the new SSL configuration.

   The new SSL configuration must be associated with *all* Web container SSL endpoints in the server, for example, `WC_defaulthost_secure` and `WC_adminhost_secure`:

   a. Select **Security** → **SSL certificate and key management** → **Manage endpoint security configurations** → **Inbound** → ***cellName*** → **nodes** → ***nodeName*** → **servers** → ***serverName*** → **WC_defaulthost_secure**.

   a. Select **Override inherited values**, and select the SSL configuration as shown in Figure 8-23.



Specific SSL configuration for this endpoint

☑ Override inherited values

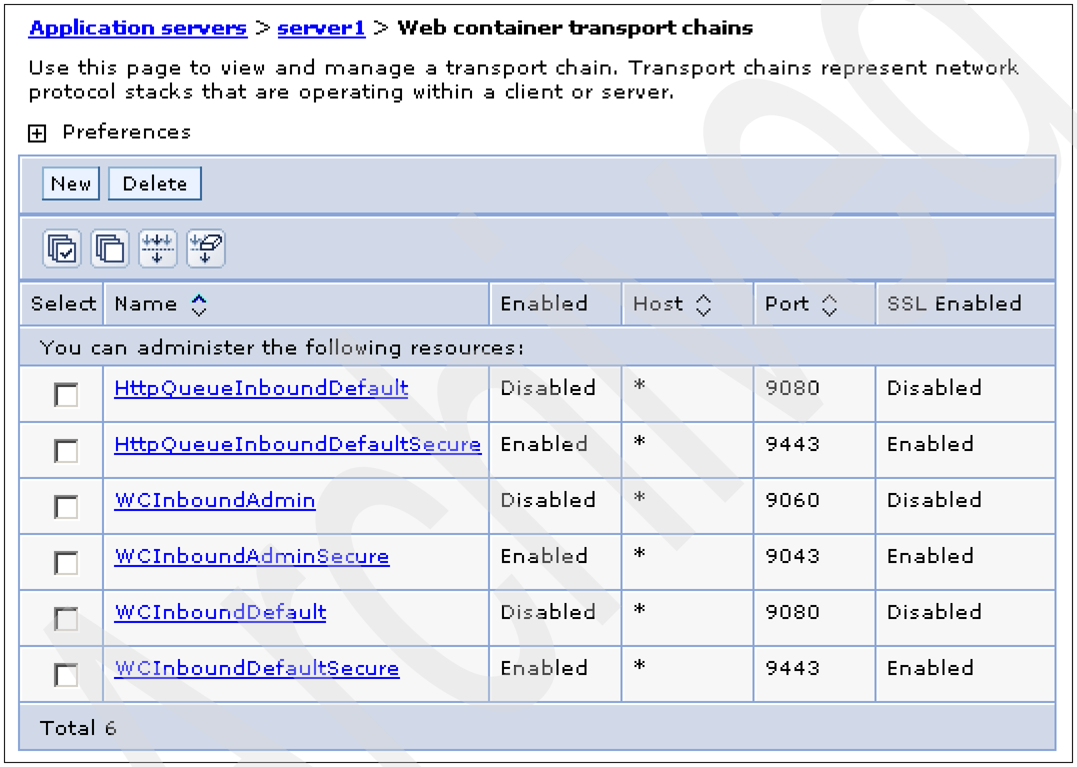
SSL configuration
ClientCertSSLSettings

Certificate alias in key store
default

Figure 8-23 Selecting the SSL configuration

Note: In a stand-alone server, if the SSL administrative port is available for internal users (not just WebSphere administrators), it has to be configured to require client authentication also.

If you cannot find the specific endpoint through the configuration tree, check Web container transport chains by using **Servers** → **Server Types** → **WebSphere Application Servers** → ***serverName*** → **Web Container Settings** → **Web container transport chains** as shown in Figure 8-24.


Application servers > server1 > Web container transport chains

Use this page to view and manage a transport chain. Transport chains represent network protocol stacks that are operating within a client or server.

Select	Name	Enabled	Host	Port	SSL Enabled
You can administer the following resources:					
☐	HttpQueueInboundDefault	Disabled	*	9080	Disabled
☐	HttpQueueInboundDefaultSecure	Enabled	*	9443	Enabled
☐	WCInboundAdmin	Disabled	*	9060	Disabled
☐	WCInboundAdminSecure	Enabled	*	9043	Enabled
☐	WCInboundDefault	Disabled	*	9080	Disabled
☐	WCInboundDefaultSecure	Enabled	*	9443	Enabled
Total 6					

Figure 8-24 Web container transport chains

6. Disable all HTTP transports on the Web container.

   By default, in the WebSphere Application Server V7, there are two HTTP transports defined: `HttpQueueInboundDefault` and `WCInboundDefault`. Disable both of them. The HTTP transports must be disabled to ensure that communication is done through the secured SSL connection.

Note: In the stand-alone server, if the HTTP administrative port is available for internal users (not just WebSphere administrators), it has to be disabled also.

8.7.3 Web server configuration

In the most common scenario, there is a Web server between the client and WebSphere Application Server. In this case, the Web server must be correctly configured to support client certificate authentication.

In the following sections, we assume that IBM HTTP Server V7 is used and that WebSphere plug-in V7 is installed. Similar steps must be performed for other vendors' servers.

Enabling SSL

The IBM HTTP Server is not configured to support SSL as shipped. For information about securing the Web server with SSL, refer to the IBM HTTP Server Information Center article, "*Securing with SSL communications*," at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_setupssl.html

Access the Web server using the HTTPs protocol to validate that SSL has been configured:

```
https://webServerHostName/
```

A sample SSL configuration, which is located in the `httpd.conf` file, is shown in Example 8-18.

Example 8-18 SSL directives for the IBM HTTP Server

```
# Enable http server to listen on 443 port
Listen *:443
# Load ssl support module
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
# Define ssl enabled virtual host
<VirtualHost *:443>
   SSLEnable
   SSLProtocolDisable SSLv2
# certificate alias
   SSLServerCert webserver
</VirtualHost>
# Specify web server keystore location
```

```
KeyFile "C:/ibm/HTTPServer/etc/webserver1.kdb"
SSLDisable
```

Configuring client authentication

The Web server must be able to accept client certificates. If the certificates are self-signed, each certificate needs to be added to the Web server keystore as a signer certificate. More often, certificates are issued by a certificate authority (CA), and only the CA certificate must be added. For simplicity, we use a self-signed certificate in this example.

Refer to “Creating a self-signed certificate for an LDAP server” on page 194 for details about creating a key database and certificates.

Follow these steps to create a sample client certificate:

1. Create a new PKCS12 key database using the IBM Key Management tool (iKeyman), which is provided with IBM HTTP Server in the `IHS_ROOT\bin` folder. Name it `wasadminPrivate.p12`.
2. Create a new self-signed certificate with a common name that matches the user ID from the WebSphere registry, for example, `wasadmin`.
3. Select the new certificate, and click **Extract Certificate**.
4. Specify the following parameters, and click **OK**:
   - Data type: Either text or binary form (both types are well known and supported)
   - Certificate file name, for example, `wasadminPublic.arm`
   - Location, for example, c:\
5. Close the key database.

Two files are created:

- `wasadminPrivate.p12`, which holds the certificate with the secret private key. This file allows you to authenticate using certificates; however, if it is lost or stolen, your identity is compromised.
- `wasadminPublic.arm`, which holds the certificate public data. This file can be safely distributed. It allows you to confirm your identity, for example, by verifying signatures that were created using your private key.

Perform the following steps to add the certificate to the signer’s certificates:

1. Start the iKeyman tool.
2. Open the Web server keystore, for example:

   `C:/ibm/HTTPServer/etc/webserver1.kdb`

3. Select **Signer certificates** from the drop-down list.
4. Click **Add** and specify the path to the certificate file (`wasadminPublic.arm`) that was generated in the previous steps. Click **OK**.
5. Close the keystore.

Troubleshooting steps: If you have problems importing a certificate, review the following issues:

- ► If the Web server certificate is signed by a CA, check if this CA's certificate is in the keystore. If not, add it *before* adding the Web server certificate.
- ► If you have a message stating "`The specified database has been corrupted`" message and the "`InvalidKeyException: Illegal key size`" message in the trace log, update the IBM Software Developer Kit (SDK) policy files:
  - – Download the files from this Web site:

    http://www.ibm.com/developerworks/java/jdk/security/60/
  - – Extract the `US_export_policy.jar` file and the `local_policy.jar` file.
  - – Copy these files to the IBM HTTP Server and WebSphere `$JAVA_HOME/jre/lib/security` directory (back up the original files).

    Important: SDK updates might overwrite these unrestricted policy files. Back up before applying fix packs, and reapply the fix packs after maintenance.
- ► To enable trace, invoke iKeyman using the following command (this command needs to be on one line):

```
gsk7ikm -Dkeyman.debug=true-Dkeyman.jnitracing=ON
-Djava.security.debug=ALL 2>ikeyman.txt
```

- ► Refer to the MustGather page for more details:

  http://www-01.ibm.com/support/docview.wss?uid=swg21202820

After the keystore is updated, the Web server configuration can be changed to support client certificate authentication:

1. Open the `httpd.conf` file and update the virtual host definition with the `SSLClientAuth` option. This option can have the following values:
   - – None: This value is the default option. The Web server does not request the client certificate. *This option cannot be used.*
   - – Optional: The server requests a client certificate; however, connection is also established if the client does not provide a certificate. This option can be used for client certificate authentication.

This option is useful when the Web server provides access to many applications and several of them do not require certificate authentication. This option is also useful if the application server is configured to allow applications to fall back to basic authentication. Refer to “Default to basic authentication when certificate authentication for the HTTPS client fails.” on page 323 for more details.

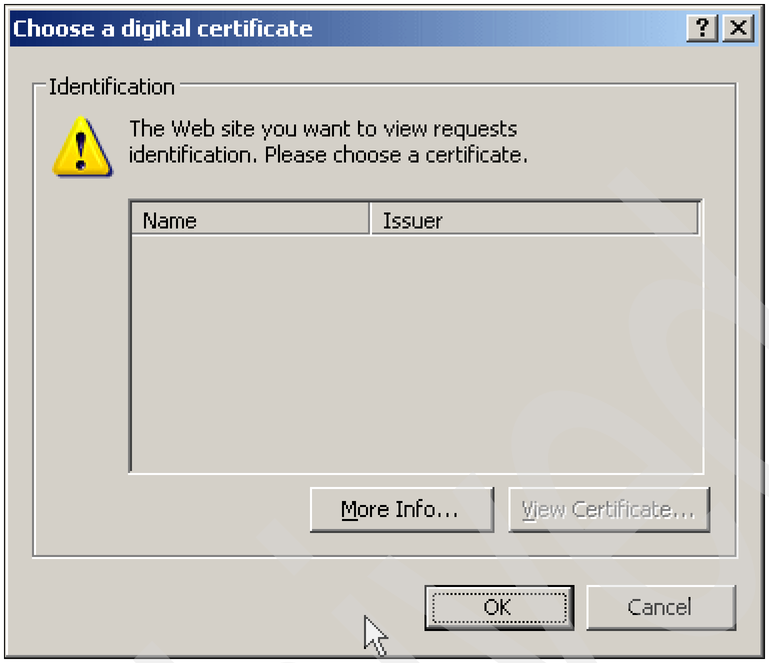
– Required: The server requires a valid certificate from the client to establish connection. This option is preferred if there is no need to support different authentication mechanisms.

The modified virtual host definition is shown in Example 8-19.

Example 8-19 Modified virtual host definition

```
<VirtualHost *:443>
   SSLEnable
   SSLProtocolDisable SSLv2
   SSLServerCert webserver
   SSLClientAuth required
</VirtualHost>
```

2. Access the Web server using `https://`*`webServerHostName`*`/` to validate that the server requests a certificate. Depending on the browser type and its configuration, access to the site is forbidden or the browser displays a dialog box asking the user of the browser to select a certificate, as shown in Figure 8-25 on page 313.



The Web site you want to view requests identification. Please choose a certificate.

Figure 8-25 Select certificate pop-up window with no valid certificates

Expect to see this pop-up window, because the Web server requests a certificate from the client at this point, and there is not a certificate in the browser.

8.7.4 Browser configuration

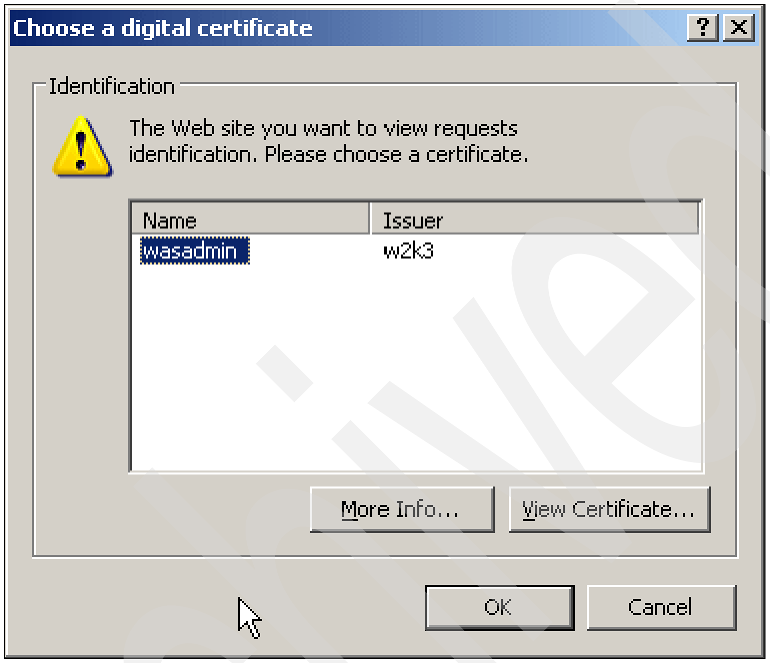
To successfully connect to an application that is secured with the client certificate, the browser must be able to use the user's certificate. Certificates are usually distributed to the users by companies that own the applications as files or smart cards. They can be provided by well known CAs or by a company CA.

We use the self-signed certificate, which is stored in `wasadminPrivate.p12`, that was created in "Configuring client authentication" on page 310.

Import the certificate to the browser:

- In Internet Explorer®, select **Tools** → **Internet Options** → **Content**, click **Certificates,** and then, click **Import**.
- In Firefox, select **Tools** → **Options** → **Advanced** → **Encryption**, click **View Certificates**, and then, click **Import**.

Access the Web server again using `https://webServerHostName/` to validate that the browser is able to use the imported certificate. Depending on the browser type and its configuration, access to the site is granted right away. Or, the browser displays a window allowing you to select the correct certificate, as shown in Figure 8-26 (Internet Explorer) or in Figure 8-27 on page 315 (Firefox).

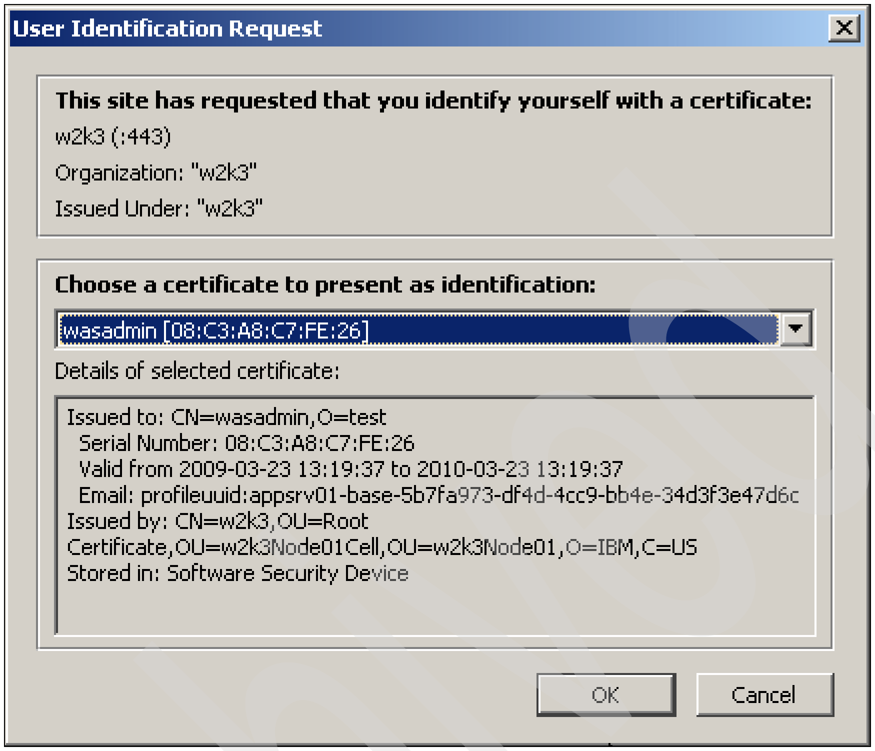

Choose a digital certificate

Identification

The Web site you want to view requests identification. Please choose a certificate.

Name	Issuer
wasadmin	w2k3

Figure 8-26 Certificate selection in Internet Explorer



This site has requested that you identify yourself with a certificate:
w2k3 (:443)
Organization: "w2k3"
Issued Under: "w2k3"

Choose a certificate to present as identification:
wasadmin [08:C3:A8:C7:FE:26]
Details of selected certificate:

```
Issued to: CN=wasadmin,O=test
  Serial Number: 08:C3:A8:C7:FE:26
  Valid from 2009-03-23 13:19:37 to 2010-03-23 13:19:37
  Email: profileuuid:appsrv01-base-5b7fa973-df4d-4cc9-bb4e-34d3f3e47d6c
Issued by: CN=w2k3,OU=Root
Certificate,OU=w2k3Node01Cell,OU=w2k3Node01,O=IBM,C=US
Stored in: Software Security Device
```

Figure 8-27 Certificate selection in Firefox

8.8 Customizing the login process

The authentication standards that are defined by the JEE specification are simple, and there is not much customization. Applications that require more sophisticated authentication methods can use the WebSphere Application Server security API, which offers greater functionality and flexibility. Because these APIs include advanced concepts, we only provide a brief description of them here. Detailed discussion about these features is beyond the scope of this publication.

We now discuss the concepts based on their complexity and the effort that is needed to develop them, from the simplest to the advanced:

- Login filter

  Use the login filter when you require minor changes to the authentication process, for example, retrieving a login exception or adding additional parameters to a user session.

The login filter is the simplest solution. It uses a servlet filter, which is mapped to the j_security_check URL. You can use a login filter for additional authentication or processing before and after authentication, as shown in Example 8-20.

Example 8-20 Login filter concept

```
public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {

   // pre login actions here
   chain.doFilter(request, myRes);
   // post login actions here
}
```

The filter must be mapped in the application's Web deployment descriptor as shown in Example 8-21.

Example 8-21 Servlet filter in the Web deployment descriptor

```
...
<filter>
   <description></description>
   <display-name>LoginFilter</display-name>
   <filter-name>LoginFilter</filter-name>
   <filter-class>
      com.ibm.itso.sample.security.filter.LoginFilter
   </filter-class>
</filter>
<filter-mapping>
   <filter-name>LoginFilter</filter-name>
   <url-pattern>/j_security_check</url-pattern>
</filter-mapping>
...
```

- Trust association interceptor

  Use a trust association interceptor (TAI) when authentication is performed by an external proxy, a custom authentication protocol (multi-phase) is required, or subject modification is required. A TAI can be used for Web authentication only.

  The trust association interceptor interface allows WebSphere Application Server to use external third-party authentication proxies. This interface must be provided by the vendor of the third-party product, or it can be implemented to match custom needs. It provides methods to intercept HTTP requests, validate the trust with a proxy, and create authenticated subjects.

The following topics in the Information Center discuss trust association usage and the interface that must be implemented:

 – http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rsec_taisubcreate.html
 – http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.javadoc.doc/web/spidocs/com/ibm/wsspi/security/tai/TrustAssociationInterceptor.html

- Java Authentication and Authorization Service login modules

  Use Java Authentication and Authorization Service (JAAS) login modules when subject modification is required, additional custom authentication is required, or identity mapping is necessary. Use this option for Remote Method Invocation (RMI) authentication. A JAAS login module is also called during propagation login.

  WebSphere Application Server supports many JAAS plug-in points. By using a custom login module, you can make additional authentication decisions or add information to the Subject to make additional, potentially finer-grained, authorization decisions inside a JEE application.

  You can obtain more information about developing custom login modules at this Web site:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_jaascustlogmod.html

- Custom registry or custom repository adapter

  Use a custom registry or custom repository adapter when applications need to use existing or unsupported user registry implementations.

  WebSphere Application Server supports the following user registries:

  – Local operating system registry
  – Stand-alone LDAP registry
  – Federated repository, which provides support for using multiple repositories and has adapters for file, LDAP, and database (proprietary schema) repositories.

  If an application needs another kind of registry, WebSphere Application Server provides two plug-in points. You can implement the UserRegistry interface to create a stand-alone custom user registry, or you can implement the Repository interface to create a custom adapter for a federated repository.

For more information about developing a stand-alone custom user registry, visit the following page:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_tbucs.html`

For more information about developing a custom repository adapter, visit this page:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rwim_dev_vmmca.html`

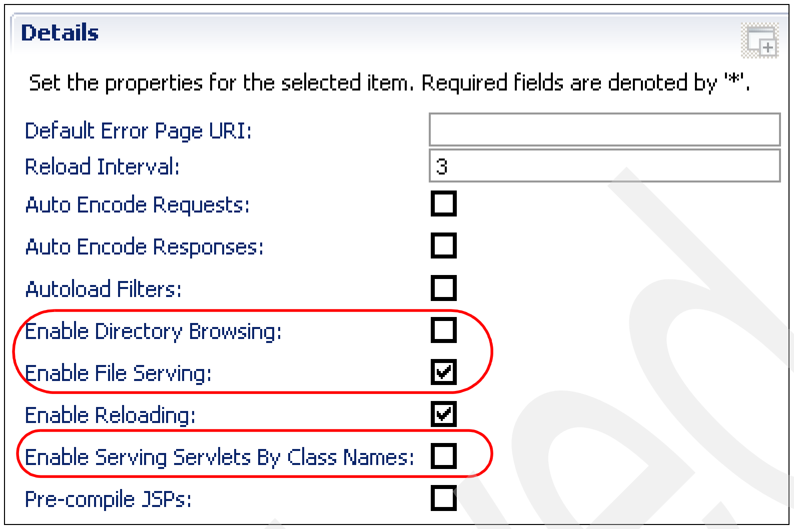
8.9 Other security-related application settings

There are a few other application-related settings to consider from a security point of view.

8.9.1 Web application extensions

There are three Web application extensions: directory browsing, file serving, and serving servlets by class name, which can be set in the Web application extension deployment descriptor (`ibm-web-ext.xml`).

To find these extensions in Rational Application Developer, open the Web module deployment descriptor and click **Open WebSphere Extensions** as shown in Figure 8-28 on page 319.



Details

Set the properties for the selected item. Required fields are denoted by '*'.

Default Error Page URI:	
Reload Interval:	3
Auto Encode Requests:	☐
Auto Encode Responses:	☐
Autoload Filters:	☐
Enable Directory Browsing:	☐
Enable File Serving:	✓
Enable Reloading:	✓
Enable Serving Servlets By Class Names:	☐
Pre-compile JSPs:	☐

Figure 8-28 Web application extensions

File serving

File serving allows WebSphere Application Server to serve static files, such as images, javascripts, and so forth, that are placed in the Web application. *Disable this setting* if you do not need to serve static files or if you plan to put them on the HTTP server.

Note: Never configure the HTTP server document root to the WAR root directory. WAR files contain application code and sensitive information that must not be served up by the Web server to users.

Directory browsing

Directory browsing allows a remote user to view the contents of the directories of the Web application. *Disable this setting* to block listing the directories.

Serving servlets by class name

This setting allows you to call servlets by their class name instead of an alias. For example, to call the servlet that is defined in the `com.ibm.itso.MyServlet`, specify a Uniform Resource Identifier (URI), such as `/servlet/com.ibm.itso.MyServlet`. *Disable this feature*. Even if your servlet URLs are secured, a malicious attacker might be able to bypass the normal URL-based security.

8.9.2 Session security integration

WebSphere Application Server by default does not authorize access to HTTP sessions. If a malicious attacker obtains the session identifier, the attacker might be able to access information in the session.

To protect applications from this kind of attack, consider enabling session security. Session security can be configured from the Web administrative console for the server (affecting all applications), the application, or the Web module:

- For the server level, select these options:

  Servers → Server Types → WebSphere Application Servers → ***serverName*** **→ Session Management**

- For the application level, select these options:

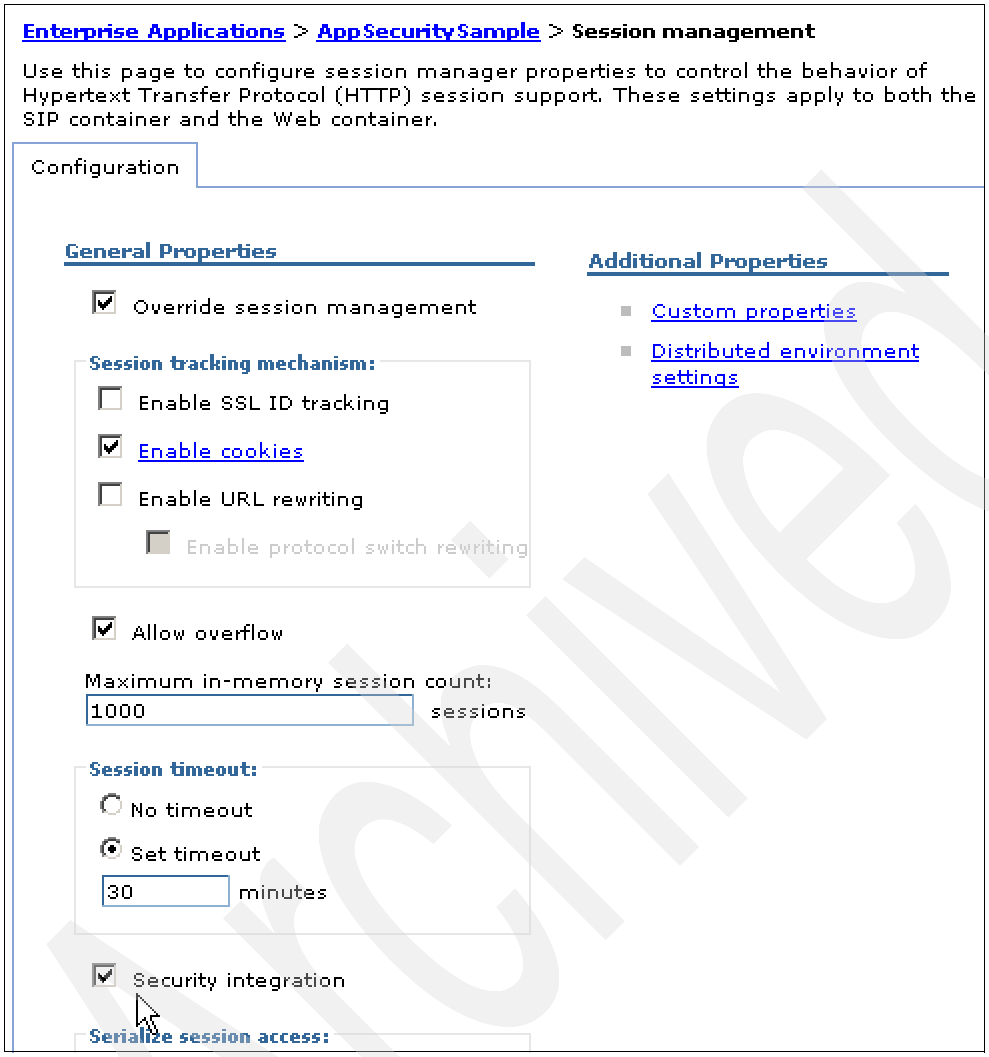
  Applications → Application Types → WebSphere Enterprise Applications → ***applicationName*** **→ Session Management**

- For the Web module level, select these options:

  Applications → Application Types → WebSphere Enterprise Applications → ***applicationName*** **→ Manage modules →** ***moduleName*** **→ Session Management**

Select **Security integration** as shown in Figure 8-29 on page 321. When this setting is enabled, the session manager associates the identity of the users with their HTTP sessions.

For the application or Web module level setting, also select **Override session management**.


Enterprise Applications > AppSecuritySample > Session management

Use this page to configure session manager properties to control the behavior of Hypertext Transfer Protocol (HTTP) session support. These settings apply to both the SIP container and the Web container.

Configuration

General Properties

[✓] Override session management

Session tracking mechanism:

[ ] Enable SSL ID tracking

[✓] Enable cookies

[ ] Enable URL rewriting

[ ] Enable protocol switch rewriting

[✓] Allow overflow

Maximum in-memory session count:
1000 sessions

Session timeout:

○ No timeout

● Set timeout

30 minutes

[✓] Security integration

Serialize session access:

Additional Properties

- Custom properties
- Distributed environment settings

Figure 8-29 Enabling session security

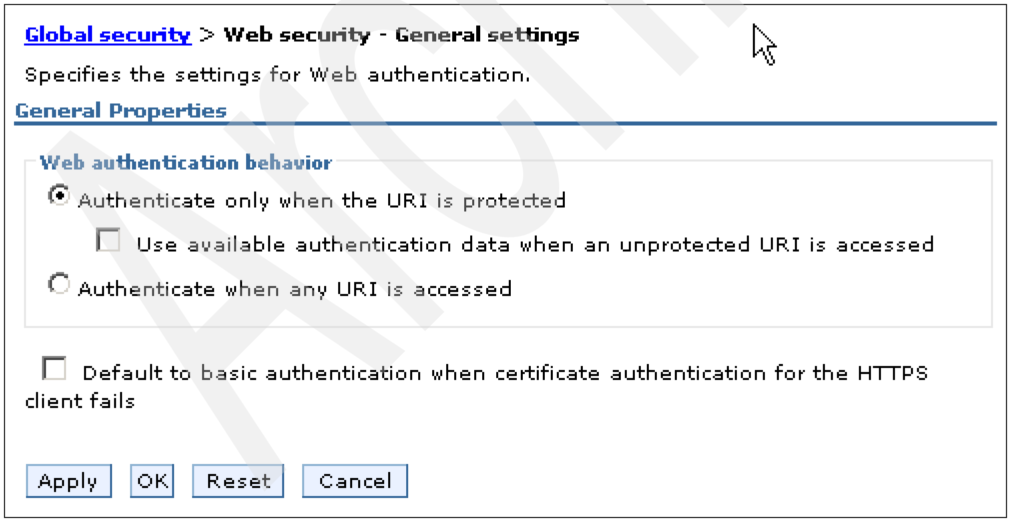
Important: After security integration is enabled, only the user who creates a session is allowed to access it. This requirement can cause an application to break if it contains a mixture of secured and unsecured pages that access the session. After a secured page or servlet accesses the session, the owner is set. If later, an insecure page or servlet tries to access the session, it will receive an authorization exception.

To allow authentication data to be available for unsecured pages, configure the Web authentication options in global security as discussed in "Modifying Web authentication behavior" on page 322. Specifically, enable **Use available authentication data when an unprotected URI is accessed**.

Modifying Web authentication behavior

WebSphere Application Server by default only requests authentication when protected resources are accessed. The authentication data can later be programmatically accessed only on secured resources. After a session is accessed from a protected page, it is not available to unprotected pages. You might have a serious issue if an application contains a mixture of protected and unprotected pages.

To change this behavior in the Web administrative console, select **Global security** → **Web and SIP security** → **General settings** as shown in the Figure 8-30.


Global security > Web security - General settings

Specifies the settings for Web authentication.

General Properties

Web authentication behavior

[✓ Authenticate only when the URI is protected]

☐ Use available authentication data when an unprotected URI is accessed

Authenticate when any URI is accessed

☐ Default to basic authentication when certificate authentication for the HTTPS client fails

Figure 8-30 Customizing authentication behavior

The Web authentication settings are listed next:

- Authenticate only when the URI is protected.

  This selection is the default option. WebSphere Application Server only requests authentication when a protected resource is accessed. Authentication data is not available for unprotected pages and servlets.

  – Use available authentication data when an unprotected URI is accessed.

    If this option is selected, authentication is still performed only when a protected resource is accessed, but once authenticated, the data is also available on unprotected pages. The developer can use the getRemoteUser, isUserInRole, and getUserPrincipal methods to retrieve an authenticated identity on these pages. The user can go back to the public part of the application and still retain authentication and session data.

- Authenticate when any URI is accessed.

  If this option is selected, authentication is performed when any application resource is accessed. It might be useful, when you want to ensure that only authenticated users access applications. For example, use this option if there is an application without support for JEE security that needs to be available only for authenticated users.

- Default to basic authentication when certificate authentication for the HTTPS client fails.

  Setting this option allows the client to log in using the user name and password if client certificate authentication fails.

  Note: This setting applies only to the application server. If client certificate authentication is configured as required at the Web server, it will not fail over to basic authentication. To achieve similar behavior with the Web server, configure the client certificate authentication as optional and not as required.

When these options are set on the Global security page, they apply to all servers in the cell. However, in certain situations, you might want to enable this behavior only for a selected application. To do this, you have to deploy the application to a separate server in the cell and override these settings using Java virtual machine (JVM) custom properties.

To specify the system property, complete the following steps:

1. Click **Servers** → **Server Types** → **WebSphere Application Servers** → ***serverName***.

2. Under Server infrastructure, click **Java and Process Management** → **Process definition**.
3. Under Additional properties, click **Java Virtual Machine** → **Custom properties** → **New**.
4. Specify the following properties from Table 8-1.

Table 8-1 Web authentication properties

Property name	Value	Description
com.ibm.wsspi.security.web.webAuthReq	lazy	Authenticate only when the URI is protected.
com.ibm.wsspi.security.web.webAuthReq	persistent	Use available authentication data when an unprotected URI is accessed.
com.ibm.wsspi.security.web.webAuthReq	always	Authenticate when any URI is accessed.
com.ibm.wsspi.security.web.failOverToBasicAuth	true	Default to basic authentication when certificate authentication for the HTTPS client fails.

9

Securing an Enterprise JavaBeans application

This chapter discusses the security aspects of Enterprise JavaBeans (EJBs) in an enterprise application. It shows how to protect applications using declarative and programmatic security. It explains how to use annotations to secure beans and how these annotations are related to the deployment descriptor. This chapter briefly discusses more advanced features, such as delegation, and the usage of the Java Authenticating and Authorization Service (JAAS).

The second part of this chapter explains how to protect access to the EJB container using transport and message layer authentication. This chapter also briefly discusses more advanced concepts, such as security attribute propagation and identity assertion.

This chapter assumes basic Java Platform, Enterprise Edition (JEE) knowledge, so we do not discuss terms, such as session beans and message-driven beans, in detail. This chapter contains the following topics:

- “Java Authentication and Authorization Service” on page 361
- “Using annotations” on page 365

9.1 Application security

A typical JEE application consists of both Web and EJB modules. EJB components implement an application's business logic and can provide services for external clients. Because EJBs have access to sensitive data and can be called by remote clients, they must be well protected.

There are two types of EJBs in EJB 3.0:

- Session beans that represent business functions and services
- Message-driven beans (MDBs) that can process asynchronous messages that are sent to the JEE application

Note: Entity beans have been replaced by lightweight Java Persistence API (JPA) entities and are not considered as Enterprise beans any more.

This chapter focuses on session beans, because they are called by clients (servlets, other EJBs, application clients, and so forth). Message-driven beans however are called by the server infrastructure and have no client credentials associated with them. To call a secured EJB from a message-driven bean, you can use delegation or the JAAS API.

9.2 Security mechanisms

Depending on the requirements, securing an application can be quite easy or extremely challenging. Consider the following options:

- If you only need to allow a certain class of user (security role) access to specific application functions (methods of the enterprise bean), you can start with 9.4, "Declarative security" on page 329.
- When declarative security alone is insufficient to fulfill the security needs of an application, you can use programmatic security to make access decisions. Refer to 9.5, "Programmatic security" on page 346.
- If you have to switch caller identity to call an EJB, or you need to call a secured bean from an unsecured one (such as an MDB), refer to 9.6, "Delegation" on page 353 and 9.7, "Java Authentication and Authorization Service" on page 361.

- Sometimes, security requirements are so complex that JEE security is not enough. In that case, consider using a third-party security framework or an external Java Authorization Contract for Containers (JACC) provider. These concepts are beyond the scope of this book and are not discussed.

When you design an application, always consider using JEE security first rather than a custom-developed framework. JEE security relies on a strong security infrastructure, is already built-in, and is enforced by the application server.

9.3 JEE Security policies: Annotations and XML descriptors

JEE security policies can be defined either through entries in the XML deployment descriptors or through annotations. *Annotations* are a new concept, which was introduced in EJB 3.0, that can simplify development. Developers can use annotations to provide assembly data inside a class file instead of in the deployment descriptor. Annotations apply to both declarative and programmatic security, although there are security aspects that can only be specified in a deployment descriptor (for example, role description). An annotation can be overridden with the deployment descriptor.

Security annotations can only be used when developing EJB 3.0 modules for WebSphere Application Server V7 or WebSphere Application Server V6.1 with the EJB 3.0 Feature Pack installed.

The following security annotations can be used by EJBs:

- RolesAllowed
- PermitAll
- DenyAll
- DeclareRoles
- RunAs

For more information about when to use annotations and the deployment descriptor, refer to 9.8, “Using annotations” on page 365.

9.4 Declarative security

Declarative security allows security polices to be configured transparently to the application code. All information is specified via metadata using annotations or EJB deployment descriptors. There are no complicated security rules hard-coded in the application code using any security API.

Security is enforced by the application server infrastructure. The EJB container retrieves the security policy from the descriptor to get the security roles that are allowed to execute bean methods. The container uses the security context that is associated with the method call to determine if the user has the required role.

Prior to EJB 3.0, security information was defined in the EJB deployment descriptor at application assembly time. Although the JEE specification designates a separate assembler role for this process, it is mostly done by developers. EJB 3.0 allows developers to put security annotations directly in the bean class to define the security policies for the bean. The application assembler can still override this information using the EJB deployment descriptor.

One of the benefits of declarative security is the ability to change application security settings according to the client's needs without changing the application code. Changes are made in the EJB deployment descriptor. All deployment descriptors can be created and modified using Rational Application Developer for WebSphere Software V7.5.

During the installation of the application, the security policies and the roles defined using annotations are combined with the security policies and the roles defined within the EJB deployment descriptor. For example, certain policies can be defined only with annotations, and other policies can only be defined using the EJB deployment descriptor. If policies overlap, those policies defined in the deployment descriptor take precedence.

9.4.1 Protecting beans using annotations

Protecting beans using annotations is quite easy. The @RolesAllowed annotation defines the list of security roles that can invoke the methods of the bean. This annotation can be placed at the class level to protect all bean methods that are part of the business interface and can also be placed at the method level where it affects only that method. An annotation placed at the method level overrides an annotation specified at the class level. Example 9-1 on page 330 shows how to use the RolesAllowed annotation at the method level.

Example 9-1 Using RolesAllowed annotation at the method level

```
import javax.annotation.security.RolesAllowed;
...

@RolesAllowed("admin")
public String methodForRole() {
   // Method that can be called by users in “admin” role
   ...
}
```

Example 9-2 shows how to use the RolesAllowed annotation at the class level with multiple security roles specified.

Example 9-2 Using RolesAllowed annotation at the class level

```
import javax.annotation.security.RolesAllowed;

@RolesAllowed({"user", "manager"})
@Stateless
public class SampleClass implements SampleInterface {
....
}
```

To specify that a method can be called by anyone, even unauthenticated users, use the @PermitAll annotation. The @PermitAll annotation is the default annotation for the bean class, so if you do not specify any annotations, all bean methods are unprotected. It can be specified at the class level or at the method level. Example 9-3 shows how to use the PermitAll annotation.

Example 9-3 Using PermitAll annotation at the method level

```
import javax.annotation.security.PermitAll;
...

@PermitAll
public String methodForAll() {
   // Method that can be called by any user even anauthenticated
   ...
}
```

Best practice: In general, avoid the PermitAll annotation, because it allows anyone to call the bean method. It is better to define a role, such as Everybody or All users, and have the ability to narrow the permissions later. For example, you can narrow the permissions later to only authenticated users or specific groups.

For the most effective usage of annotations, you can specify @RolesAllowed annotations at various levels. If most of methods in the bean are for a specific role, define it at the class level and then override permissions at the method level, as shown in Example 9-4.

Example 9-4 Combining annotations

```
// Default role for all methods in the class
@RolesAllowed("user")
@Stateless
public class OverrideTester implements OverrideTesterLocal {

   // no annotation - uses class default - “user” role
   public void aMethod() { ... }

   // annotation overrides class default - mehtod for “admin” role
   @RolesAllowed("admin")
   public void bMethod() { ... }
```

If you must exclude a particular method from execution, use the @DenyAll annotation. This annotation can be placed at the method level only. It overrides annotations defined at the class level. Example 9-5 shows how to use the DenyAll annotation.

Example 9-5 Using DenyAll annotation at the method level

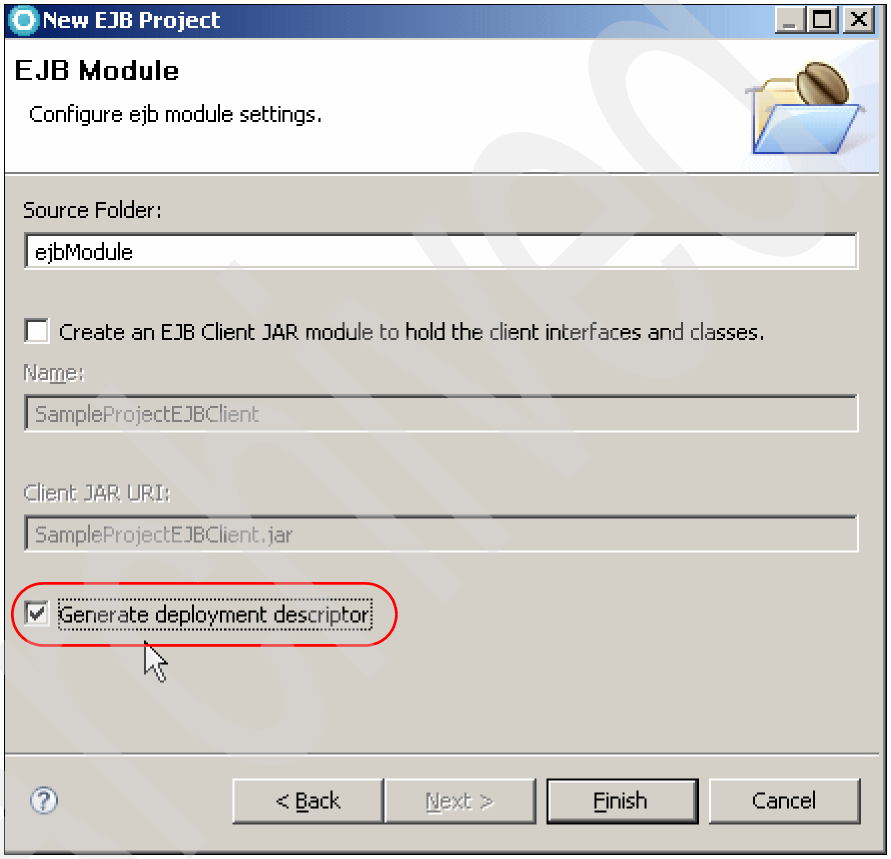
```
import javax.annotation.security.RolesAllowed;
...

@DenyAll
public String methodForNobody() {
   // method cannot be executed by any role - excluded
   ...
}
```

9.4.2 Protecting beans using the deployment descriptor

Protecting beans using the XML deployment descriptor is slightly more complicated. First, you need a deployment descriptor. There a few ways to get one:

- If you are creating a new EJB Project in Rational Application Developer, select **Generate deployment descriptor** in the wizard, as shown in Figure 9-1.



Source Folder: ejbModule

[ ] Create an EJB Client JAR module to hold the client interfaces and classes.

Name: SampleProjectEJBClient

Client JAR URI: SampleProjectEJBClient.jar

[✓] Generate deployment descriptor

Figure 9-1 Create project with deployment descriptor

- For existing EJB projects, right-click the project name, and then, select **Java EE** → **Generate Deployment Descriptor Stub** as shown in Figure 9-2 on page 333.

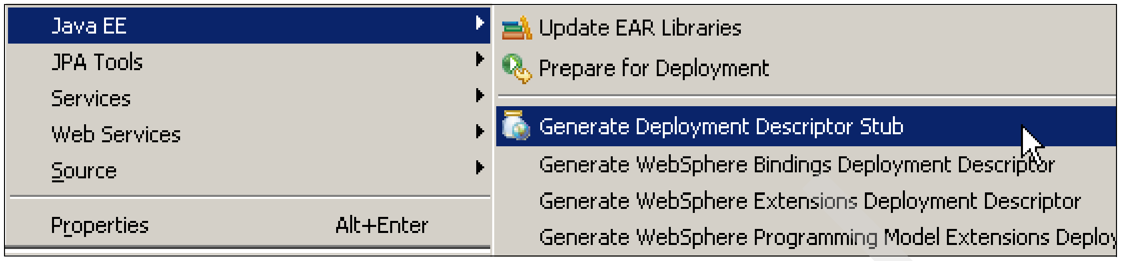

Java EE
JPA Tools
Services
Web Services
Source
Properties Alt+Enter
Update EAR Libraries
Prepare for Deployment
Generate Deployment Descriptor Stub
Generate WebSphere Bindings Deployment Descriptor
Generate WebSphere Extensions Deployment Descriptor
Generate WebSphere Programming Model Extensions Deploy

Figure 9-2 Generating deployment descriptor

Example 9-6 shows the generated, empty deployment descriptor.

Example 9-6 Generated deployment descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd" version="3.0">
   <display-name>
   AppSecuritySampleEJB</display-name>
</ejb-jar>
```

- Or, you can manually create an XML file named `ejb-jar.xml` in the `ejbModule\META-INF` folder.

After you have generated the deployment descriptor, you can start to secure the beans.

Defining security roles

JEE uses a role-based security model, which means that access to application resources is granted based on the security role. The *security role* is a logical grouping of principals. A *principal* is an element that can be authenticated, typically a user. When an application is deployed, security roles are mapped to actual users or groups from the application server's security realm.

When the security policy is defined using the deployment descriptor, security roles are the first elements that need to be defined. The following steps describe how to create a security role using Rational Application Developer for WebSphere Software V7.5:

1. In the Enterprise Explorer view, double-click the application's EJB module to expand it.

2. Double-click the EJB deployment descriptor. The EJB Deployment Descriptor Editor opens. Select the **Design** tab.
3. Select **EJB Project (*EJBModuleName*)**, and click **Add**.
4. Select **Assembly** as shown in Figure 9-3, and click **OK**. A new Assembly element is added under the EJB Project element.

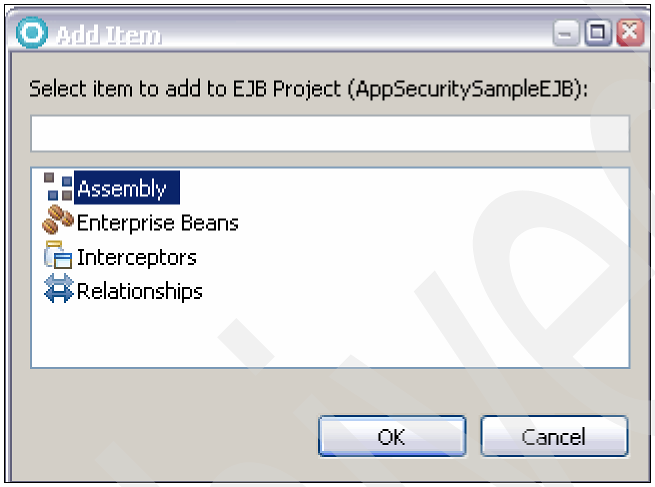
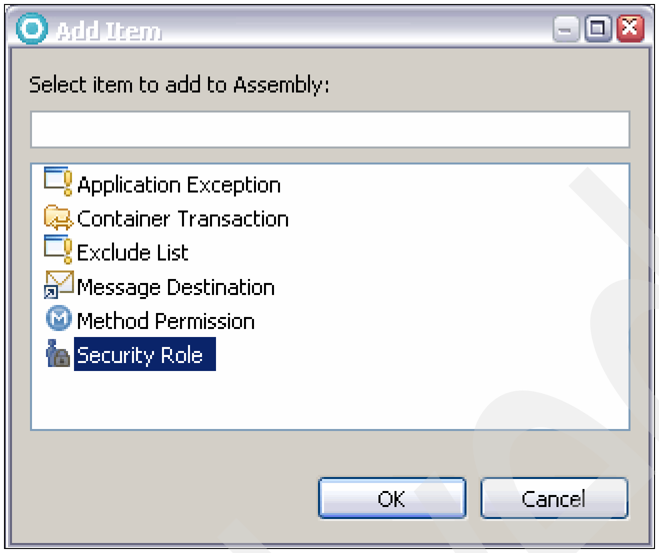


Figure 9-3 Adding the Assembly element

5. Select **Assembly**, and click **Add**.
6. Select **Security Role** as shown in Figure 9-4 on page 335, and click **OK**.

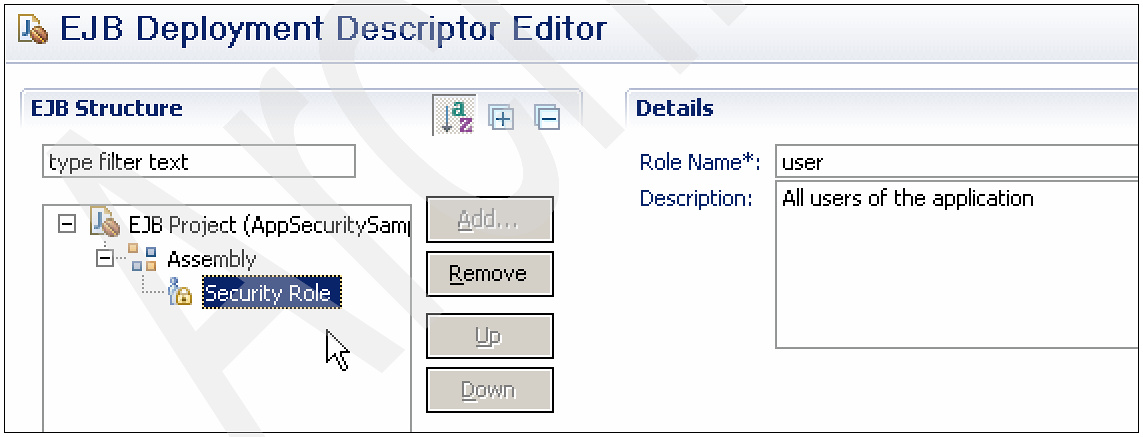


Select item to add to Assembly:

Application Exception
Container Transaction
Exclude List
Message Destination
Method Permission
Security Role

Figure 9-4 Adding Security Role element

7. The Add Security Role pop-up window opens. Enter `user` in the Security Role field and `All users of the application` in the Description field. Click **OK**. The new Security Role element is added under the Assembly element. The editor looks like Figure 9-5.



Role Name*: user
Description: All users of the application

Figure 9-5 EJB Deployment Descriptor Editor

8. Repeat steps 5-7 to add the `admin` security role.
9. Save the file using Ctrl+S.

New `<security-role>` elements are added to the assembly section of the EJB deployment descriptor. Refer to Example 9-7. Select the **Source** tab in the editor to see the entire XML file.

Example 9-7 Security Role element in the EJB deployment descriptor

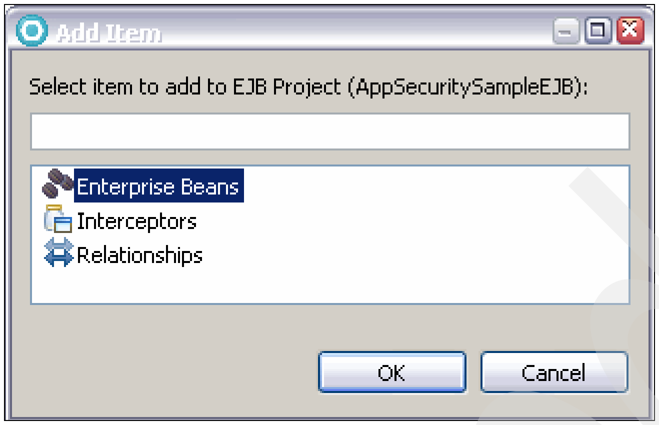
```
<assembly-descriptor>
      <security-role>
         <description>Administrators</description>
         <role-name>admin</role-name>
      </security-role
      <security-role>
         <description>All users of the application</description>
         <role-name>user</role-name>
      </security-role>
   </assembly-descriptor>
```

After security roles are defined, you can use them to grant access to the bean.

Adding enterprise beans

The next step is to add the definitions of the beans that will be secured to the deployment descriptor. The following actions describe this process:

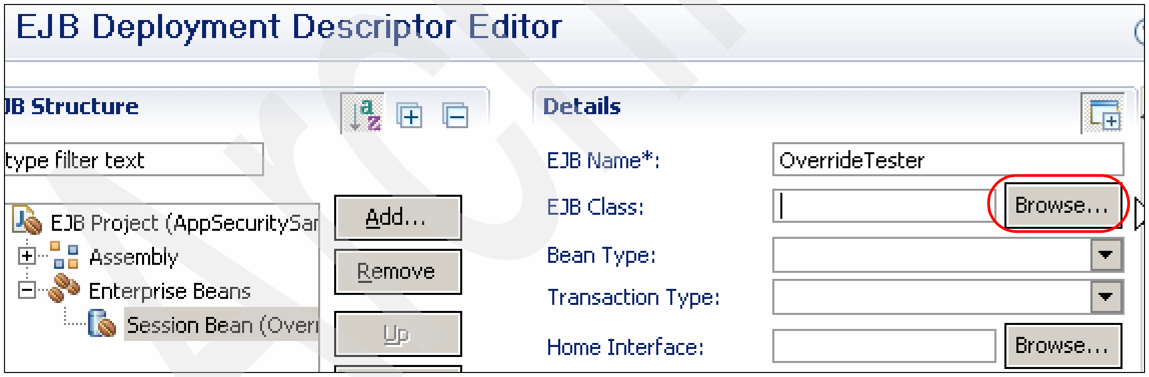
1. In the Enterprise Explorer view, double-click application's EJB module to expand it.
2. Double-click the EJB deployment descriptor. The EJB Deployment Descriptor Editor opens. Select the **Design** tab.
3. Select **EJB Project *(EJBModuleName)***, and click **Add**.
4. Select **Enterprise Beans** as shown in Figure 9-6 on page 337, and click **OK**.



Select item to add to EJB Project (AppSecuritySampleEJB):

Figure 9-6 Adding Enterprise Beans element

5. Select **Enterprise Beans,** and click **Add**.
6. Select **Session bean**, and click **OK**.
7. The New Enterprise Bean pop-up window opens. In the EJB name field, enter the bean name, for example, `OverrideTester`, and click **OK**.
8. Select the new Session Bean element (if not already selected), and click **Browse** to specify the EJB class as shown in Figure 9-7.



EJB Name*: OverrideTester
EJB Class:
Bean Type:
Transaction Type:
Home Interface:

Figure 9-7 Session bean details

9. In the Type Selection dialog, start typing the class name, and then, select **OverrideTester** from the Matching items list as shown in Figure 9-8 on page 338. Click **OK**.

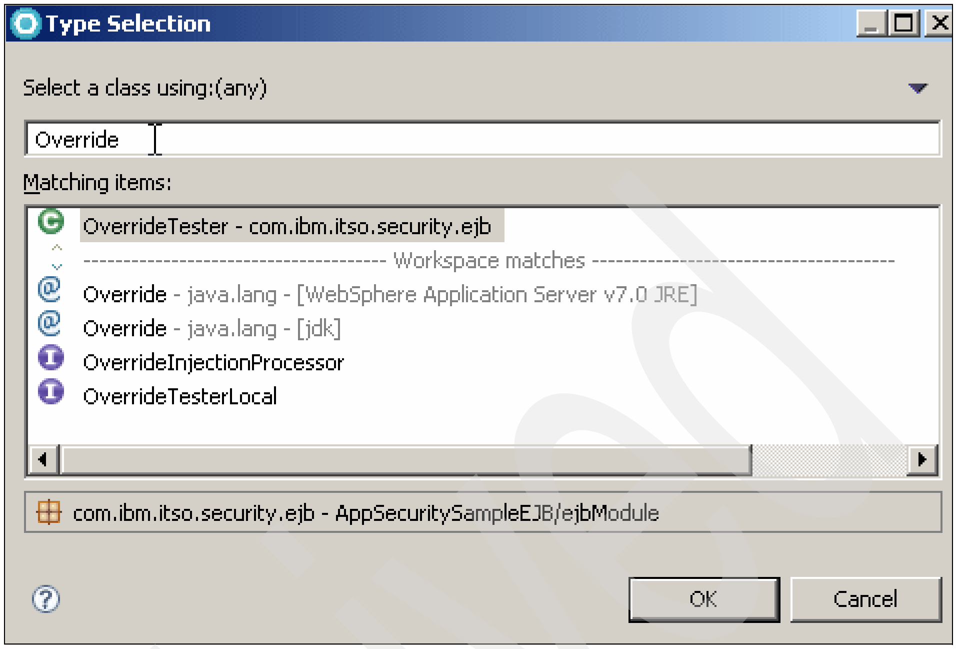

Type Selection
Select a class using:(any)
Override
Matching items:
OverrideTester - com.ibm.itso.security.ejb
Workspace matches
Override - java.lang - [WebSphere Application Server v7.0 JRE]
Override - java.lang - [jdk]
OverrideInjectionProcessor
OverrideTesterLocal
com.ibm.itso.security.ejb - AppSecuritySampleEJB/ejbModule
OK
Cancel

Figure 9-8 Selecting the EJB class

10. Save the file (Ctrl+S).

A new <enterprise-bean> element is added to EJB deployment descriptor. Refer to Example 9-8.

Example 9-8 Enterprise bean element in the EJB deployment descriptor

```
<enterprise-beans>
   <session>
      <ejb-name>OverrideTester</ejb-name>
      <ejb-class>com.ibm.itso.security.ejb.OverrideTester</ejb-class>
   </session>
</enterprise-beans>
```

Now, with the security roles and the enterprise bean defined, you can start to define method permissions.

Defining method permissions

The access policy for the bean methods is defined using the `<method-permission>` element of the EJB deployment descriptor. This element associates the security roles with the set of bean methods that can be invoked by these security roles.

The following steps describe how to define method permission:

1. In the Enterprise Explorer view, double-click the application's EJB module to expand it.
2. Double-click EJB deployment descriptor. The EJB Deployment Descriptor Editor opens. Select the **Design** tab.
3. Expand **EJB Project *(EJBModuleName)***, select **Assembly**, and click **Add**.
4. Select **Method Permission** as shown in Figure 9-9, and click **OK**.

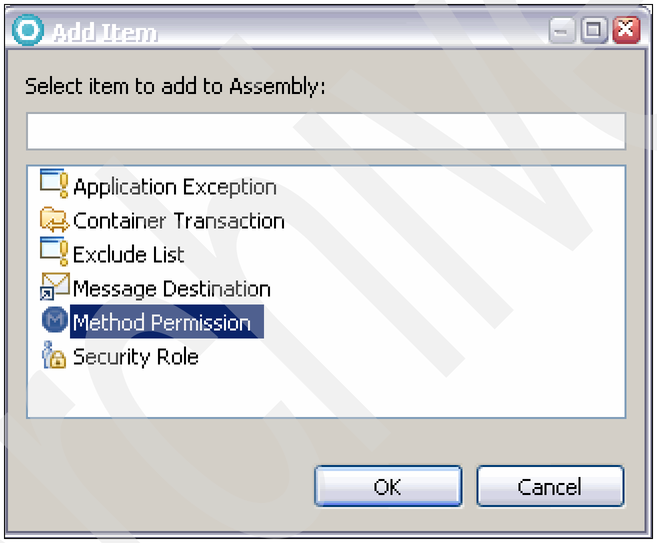
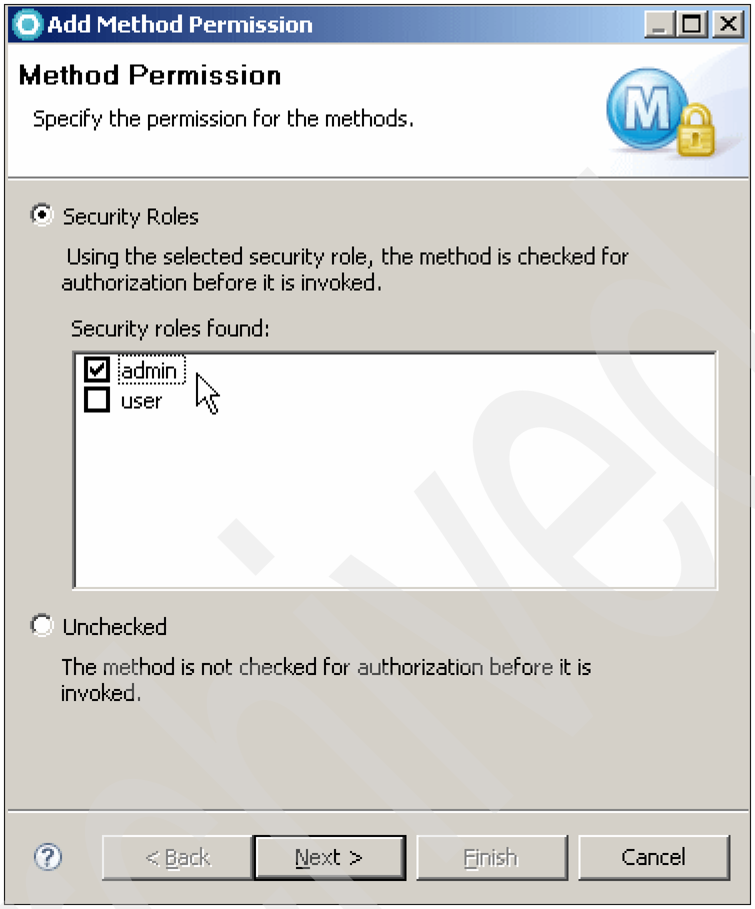


Figure 9-9 Adding Method Permission

5. The Add Method Permission wizard opens. The first pane allows you to choose the security roles that will be authorized to execute methods defined by this method permission element.

   Select the `admin` security role as shown in Figure 9-10 on page 340, and click **Next**. The **unchecked** option is described in "The unchecked option" on page 344.



Method Permission

Specify the permission for the methods.

Security Roles [✓]

Using the selected security role, the method is checked for authorization before it is invoked.

Security roles found:

- [✓] admin
- [ ] user

Unchecked

The method is not checked for authorization before it is invoked.

Figure 9-10 Selecting security roles

6. The next pane allows you to select which enterprise beans will be secured by this method permission. Select the bean as shown in Figure 9-11 on page 341, and click **Next** to see the selected bean methods.

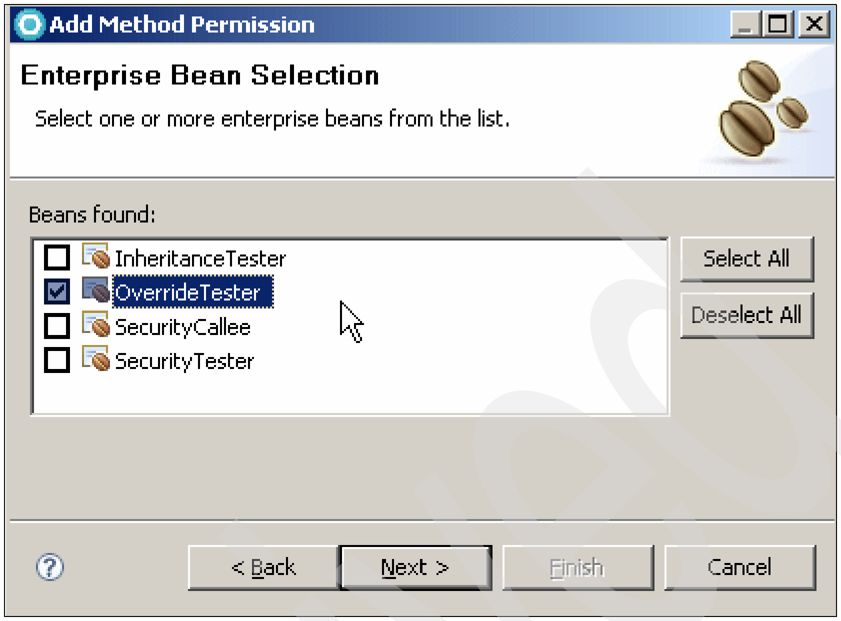


Figure 9-11 Selecting beans

7. On the Method Element pane, select the methods that will be accessible by selected security roles. You can select any subset of methods by checking the required method names, or you can check the asterisk character (*) to select all methods. For this example, select **dMethod** as shown in Figure 9-12 on page 342, and click **Finish**.

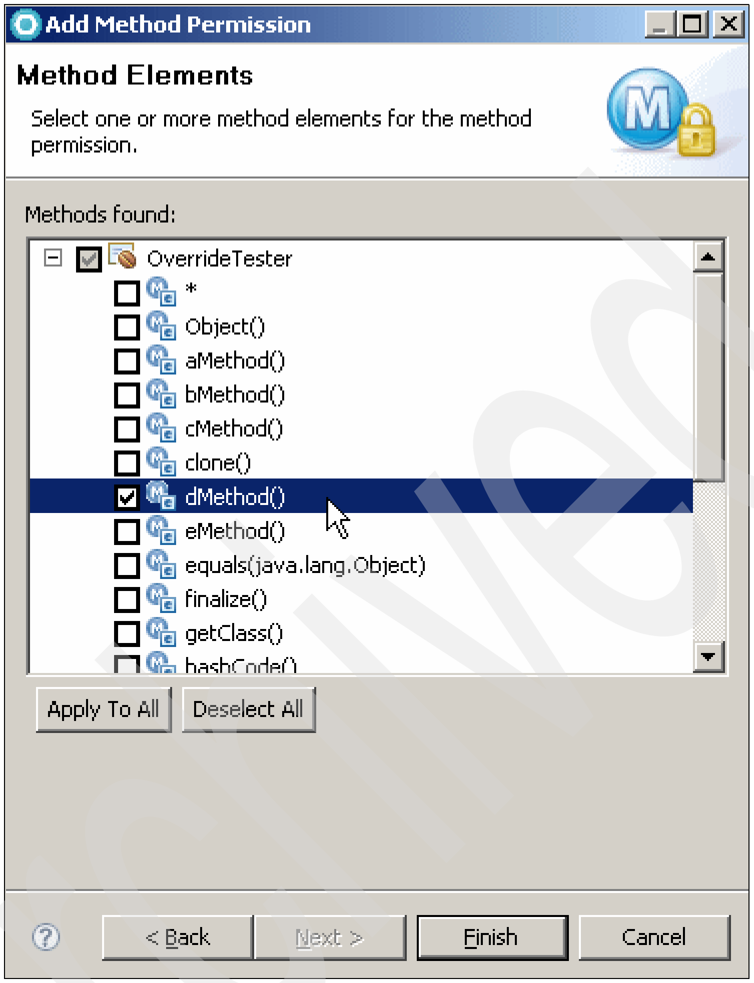


Figure 9-12 Selecting methods

8. Save the file.

A new `<method-permission>` element is added to the assembly section of the EJB deployment descriptor. Refer to Example 9-9.

Example 9-9 Method permission element in the EJB deployment descriptor

```
<method-permission>
         <role-name>admin</role-name>
         <method>
            <ejb-name>OverrideTester</ejb-name>
            <method-name>dMethod</method-name>
            <method-params>
               <method-param></method-param>
            </method-params>
         </method>
      </method-permission>
```

We have only secured access for dMethod. All of the other methods are not secured and can be called by anyone, unless there are security annotations defined in the bean class. You must define method permissions for the other methods in a similar way.

Best practice: Always protect access to all of the business interface methods.

The previous example used the most detailed method element definition: consisting of ejb-name, method-name, method-params, and method-param elements. It is best used when security settings must be defined or overridden for specific methods. If all of the methods of the bean have the same security settings, the wildcard notation with the asterisk character (*) can be used, as shown in Example 9-10. In this example, all methods of the OverrideTester bean are accessible only to users in the admin role.

Example 9-10 Using wildcard in method permission element

```
<method-permission>
         <role-name>admin</role-name>
         <method>
            <ejb-name>OverrideTester</ejb-name>
            <method-name>*</method-name>
         </method>
</method-permission>
```

If a bean has many client views (for example, remote, local, and Web service endpoint), and several of them require a separate security policy, you can use the optional <method-intf> element. This element has to be placed after the <ejb-name> element and allows you to specify interface type (valid values are: Home, Remote, LocalHome, Local, ServiceEndpoint). Example 9-11 shows how to secure methods in the remote business interface.

Example 9-11 Using method-intf to secure methods in the remote interface

```
<method-permission>
         <role-name>admin</role-name>
         <method>
            <ejb-name>SomeBean</ejb-name>
            <method-intf>Remote</method-intf>
            <method-name>*</method-name>
        </method>
</method-permission>
```

Troubleshooting tip: To diagnose problems related with overriding annotations with the EJB deployment descriptor, set the trace to:
`com.ibm.ws.security.role.metadata.*=all`

The unchecked option

Defining a method as *unchecked* allows anyone to invoke it. Carefully consider if you want to use this option.

One possible scenario is when you have many methods in the bean and only a few of them are unprotected. In this situation, you can define one method permission for all methods using a wildcard and a second for the unchecked methods.

Another case might be if you have beans with security annotations and plan to to override them with the deployment descriptor. The use of this technique, however, must be carefully considered. If the application is not prepared to be called by anonymous users, you can break the application.

Specifying method permission with the unchecked option is shown in Example 9-12. As you can see, there are no <*role-name*> elements in this method permission.

Example 9-12 Method permission with unchecked option

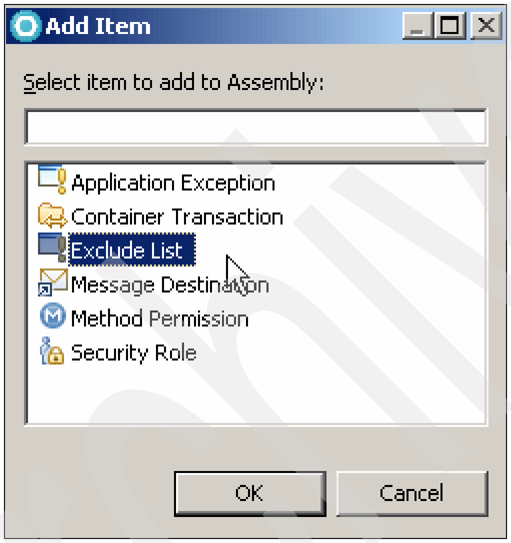
```
<method-permission>
        <unchecked />
        <method>
            <ejb-name>OverrideTester</ejb-name>
            <method-name>eMethod</method-name>
            <method-params>
                <method-param></method-param>
            </method-params>
        </method>
</method-permission>
```

Excluding methods from execution

If you want to deny access to a specific method, you can add it to the exclude list. Methods in the exclude list cannot be invoked by any user in any role.

To add a method to the exclude list, follow these steps:

1. In the Enterprise Explorer view, double-click the application's EJB module to expand it.
2. Double-click the EJB deployment descriptor to open it. Select the **Design** tab.
3. Expand **EJB Project *(EJBModuleName)***, select **Assembly**, and click **Add**.
4. Select **Exclude List** as shown in Figure 9-13, and click **OK**.


Add Item

Select item to add to Assembly:

Application Exception
Container Transaction
Exclude List
Message Destination
Method Permission
Security Role

Figure 9-13 Adding Exclude List element

5. Select the bean, and click **Next**.
6. Select the methods to exclude, and click **Next**.
7. Click **Finish** to end the wizard. Save the EJB deployment descriptor.

A new `<exclude-list>` element is added to the assembly section of the EJB deployment descriptor. Refer to Example 9-13 on page 346.

Example 9-13 Exclude list element in the EJB deployment descriptor

```
<exclude-list>
        <method>
           <ejb-name>DescriptorTester</ejb-name>
           <method-name>methodExcluded</method-name>
           <method-params>
              <method-param></method-param>
           </method-params>
        </method>
</exclude-list>
```

When you attempt to invoke a method that is in the exclude-list, you get the exception that is shown in the Example 9-14.

Example 9-14 Exception while executing method that is in the exclude list

```
[2/24/09 16:33:49:796 CET] 00000026 SecurityColla A   SECJ0053E: 
Authorization failed for defaultWIMFileBasedRealm/joe while invoking 
(Bean)AppSecuritySample#AppSecuritySampleEJB.jar#DescriptorTester 
methodExcluded::3 
AppSecuritySample:AppSecuritySampleEJB.jar:DescriptorTester:methodExclu
ded::3 is excluded
```

After security settings for the application are defined, you have to map users from the user registry to the security roles. You can use either Rational Application Developer for WebSphere Software V7.5 or the application server during or after the application installation (refer to 7.2, "Deploying a secured enterprise application" on page 254).

9.5 Programmatic security

Programmatic security allows the developer to get details about the authenticated user, including the user's identity and security role. This type of security is useful when declarative security is not enough to fulfill the application security requirements.

9.5.1 Security API

The javax.ejb.EJBContext interface provides two methods that allow the bean provider to access security information about the enterprise bean caller:

- java.security.Principal getCallerPrincipal(): This method allows the developer to get the name of the caller.
- Boolean isCallerInRole(Sting roleNameReference): This method allows the developer to test if the calling user has the specific role.

To use these methods, you first need to get the object that implements the javax.ejb.SessionContext interface. The SessionContext interface extends the EJBContext interface and has additional methods that might be useful for the session bean developer. Use resource injection to initialize the SessionContext variable as shown in Example 9-15.

Example 9-15 Injecting SessionContext

```
@Stateless
public class SecurityTester implements SecurityTesterLocal {

   @Resource
   SessionContext ctx;
...
}
```

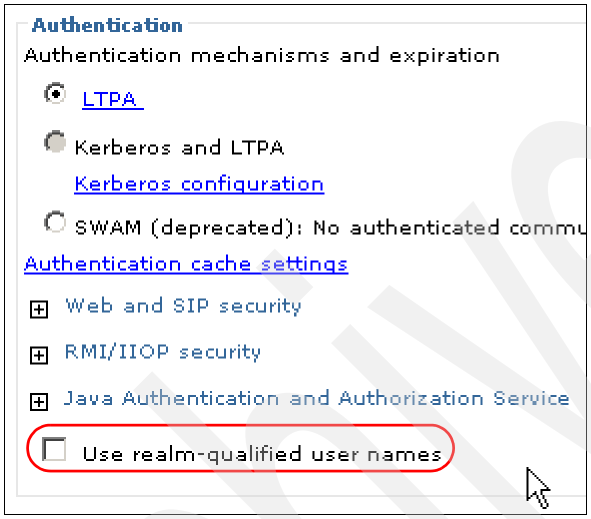
The getCallerPrincipal method can be used to get the name of the calling user as shown in Example 9-16. Later, you can store the user name that was obtained in a log file or a database to provide auditing for user actions in the application.

Example 9-16 The getCallerPrincipal method

```
...
   // get the caller principal
   Principal principal = ctx.getCallerPrincipal();
   // print caller name to system out
   System.out.println("methodX called by: " + principal.getName() );
```

Note: The getCallerPrincipal method, according to the specification, cannot return null. If the bean is called by an unauthenticated user, the principal.getName() method will return the name “UNAUTHENTICATED”.

Be aware that the format of the principal name is not defined in the specification, and it is not standardized. It can vary depending on user registry settings and application server security settings. For example, if an application server is configured to return realm-qualified user names, getCallerPrincipal returns a user name prefixed with the realm name. For example, it might return `defaultWIMFileBasedRealm/gas` instead of `gas`. You can configure this option using the administrative console in the Global Security page as shown in Figure 9-14.



Authentication

Authentication mechanisms and expiration

● LTPA

○ Kerberos and LTPA

Kerberos configuration

○ SWAM (deprecated): No authenticated commu

Authentication cache settings

⊞ Web and SIP security

⊞ RMI/IIOP security

⊞ Java Authentication and Authorization Service

☐ Use realm-qualified user names

Figure 9-14 Realm-qualified user names option

The isCallerInRole method allows the developer to make access decisions in the method body based on the user's role. For example, users with a specific role can accept credit requests with a higher amount, and so forth. Example 9-17 shows how to use this method.

Example 9-17 Using isCallerInRole method

```
// check if user is in role
 if(ctx.isCallerInRole("manager")) {
   System.out.println("You are a member of manager role");
      // ... do something for managers here
 }
 else {
      // ordinary users go here
 }
```

The parameter in the isCallerInRole method is a security role reference. This reference has to be defined either through annotations or in the EJB deployment descriptor. If there is no reference with the given name specified, the container assumes that the parameter is a security role name and checks if the user is in this role. If the user is not in the specified role or there is no role with given name, the method returns `false`.

Defining security role references using annotations

A security role reference can be defined using the @DeclareRoles annotation. The DeclareRoles annotation can be specified at the class level only and is used to define roles that are tested by calling isCallerInRole.

Example 9-18 shows how to define a role reference using the DeclareRole annotation and then how to use it later in the bean method.

Note: In the isCallerInRole method, you can use any of the references and roles that are defined by:

- DeclareRole annotations
- RolesAllowed annotations, even if they are used in different beans
- Security role references that are defined in the EJB deployment descriptor
- Security roles that are defined in the EJB deployment descriptor

Do not repeat values from RolesAllowed or the EJB deployment descriptor in the DeclareRoles annotation.

Example 9-18 DeclareRoles definition and usage sample

```
// Declaration of the manager role reference
@DeclareRoles("manager")
@Stateless
public class SecurityTester implements SecurityTesterLocal {

   // We need to inject SessionContext to check the role assignment
   @Resource
   SessionContext ctx;

   public void testRoles() {
      // role defined by RolesDefined annotation
      if(ctx.isCallerInRole("manager")) {
         System.out.println("You are a member of manager role");
      }
      // role defined by RolesAllowed annotation
      if(ctx.isCallerInRole("admin")) {
            System.out.println("You are a member of admin role");
        }
        ....
    }

    @RolesAllowed(“admin”)
    public String methodForRole() {
     ... }
}
```

Any roles that are defined using the DeclareRole annotation must be mapped to users or groups from the user registry. Map them using Rational Application Developer for WebSphere Software V7.5 or on the application server during or after the application installation (refer to Chapter 7, “Application security” on page 253).

Defining security role references

Security role references are used to separate the role names that are hardcoded in the bean class by the developer from the roles that are defined at assembly. At assembly, you can link role references to the security roles defined in the application deployment descriptor. With EJB 3.0 and security annotations, role names are hardcoded in the bean class anyway, so the usage of references in the EJB deployment descriptor is less important, however, still possible.

The following steps describe how to create a security role reference and link it to a security role using the EJB deployment descriptor. We assume that the EJB deployment descriptor is already generated. Follow these steps:

1. In the Enterprise Explorer view, double-click application’s EJB module to expand it.
2. Double-click the EJB deployment descriptor to open it. Select the **Design** tab.
3. Expand **EJB Project *(EJBModuleName)*** → **Enterprise Beans**. Add a new enterprise bean with the name `SecurityTester` and the EJB class `com.ibm.itso.security.ejb.SecurityTester` (similar to “Adding enterprise beans” on page 336).
4. Select **SecurityTester bean**, and click **Add**.
5. Select **Security Role Reference** as shown in Figure 9-15 on page 351 and click **OK**.

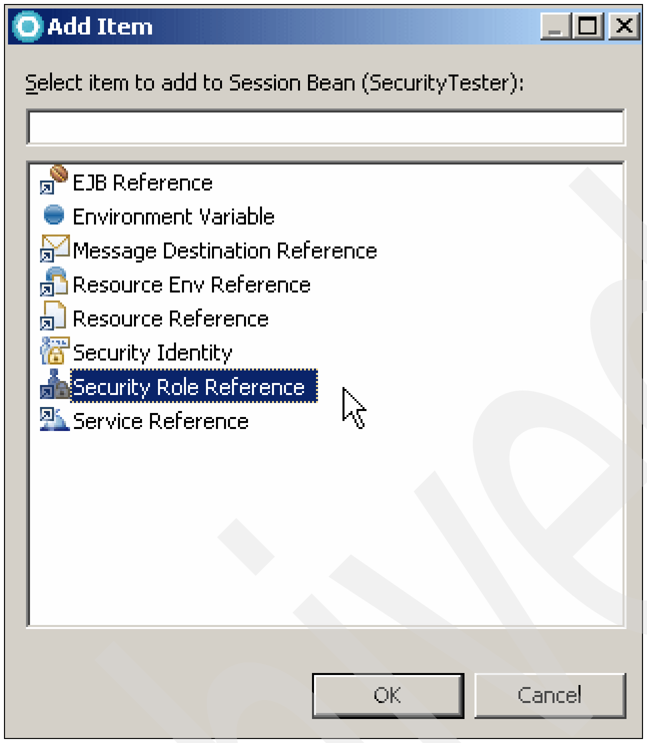
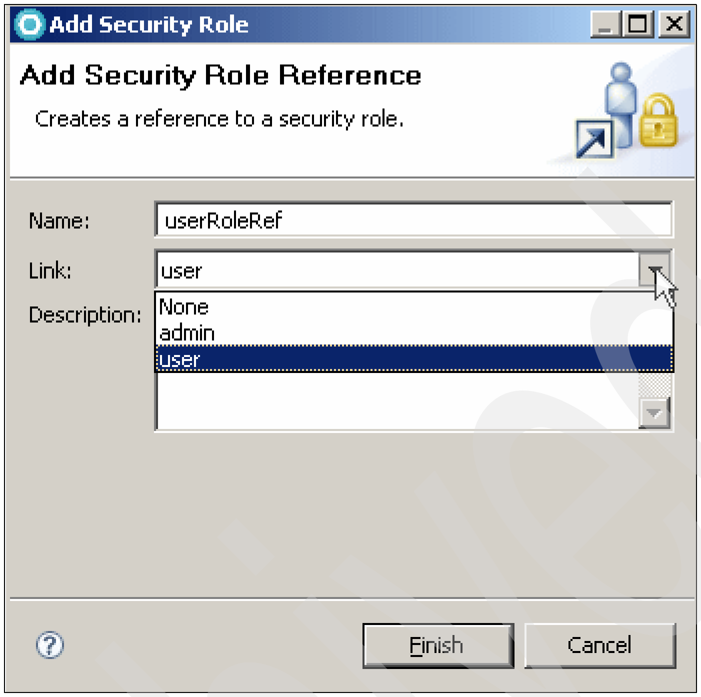


Figure 9-15 Adding Security Role Reference

6. The Add Security Role pop-up window opens. Specify the role reference used in the code in the Name field, and link the reference to the security role using the Link field as shown in Figure 9-16 on page 352. If you do not have any security roles defined, refer to "Defining security roles" on page 333 for information about how to define one.


Add Security Role

Add Security Role Reference

Creates a reference to a security role.

Name: userRoleRef

Link: user

Description:

Figure 9-16 Defining and linking security reference

Click **Finish**.

7. Save the EJB deployment descriptor.

A new `<security-role-ref>` element is added to the SecurityTester bean definition in the EJB deployment descriptor. It contains the reference name and the role name to which the reference is linked. Refer to Example 9-19.

Example 9-19 Security reference in the EJB deployment descriptor

```
<session>
   <ejb-name>SecurityTester</ejb-name>
   <ejb-class>com.ibm.itso.security.ejb.SecurityTester</ejb-class>
   <security-role-ref>
      <description></description>
      <role-name>userRoleRef</role-name>
      <role-link>user</role-link>
   </security-role-ref>
</session>
```

9.6 Delegation

When a bean calls a method in another bean, the identity of the first caller is, by default, propagated to the next. In this way, all EJB methods in the calling chain see the same principal if they were to call the getCallerPrincipal() method. Occasionally, however, it is desirable for one EJB to call another EJB with a previously defined identity, for instance, one that is a member of a specific role.

For example, consider the message-driven bean's onMessage() method, which calls a protected method in a session bean. Because message-driven beans' onMessage() methods are executed with no caller identity, this method cannot call protected session beans. By delegating the onMessage() method to run as a role that is allowed to invoke methods of the session bean, the onMessage() method can successfully access the protected method.

9.6.1 Bean-level delegation

The EJB specification allows delegation only at the bean level. Therefore, all methods of the given bean will call other beans under a specific *RunAs* role. This role can be defined either by using security annotation or the deployment descriptor. After the RunAs role is defined, it must be mapped to a real user that is a member of the specified role. You can map it in Rational Application Developer for WebSphere Software V7.5 or on the application server during or after the application installation (refer to Chapter 7, "Application security" on page 253). All calls made by the delegated bean are done using the identity of the mapped user.

Figure 9-17 on page 354 shows EJB delegation in contrast to the default Run-As Caller mode. Follow these steps:

- In the top scenario, the identity of the caller, `caller01`, is propagated from EJB1 to EJB2.
- In the bottom scenario, EJB1 is delegated to run as `role01`. During run-as mapping, another user, `caller02`, is mapped to `role01`, and therefore, it is effectively `caller02` that calls EJB2. If EJB2 were to call EJB3, EJB3 also appears to have been called by `caller02`.

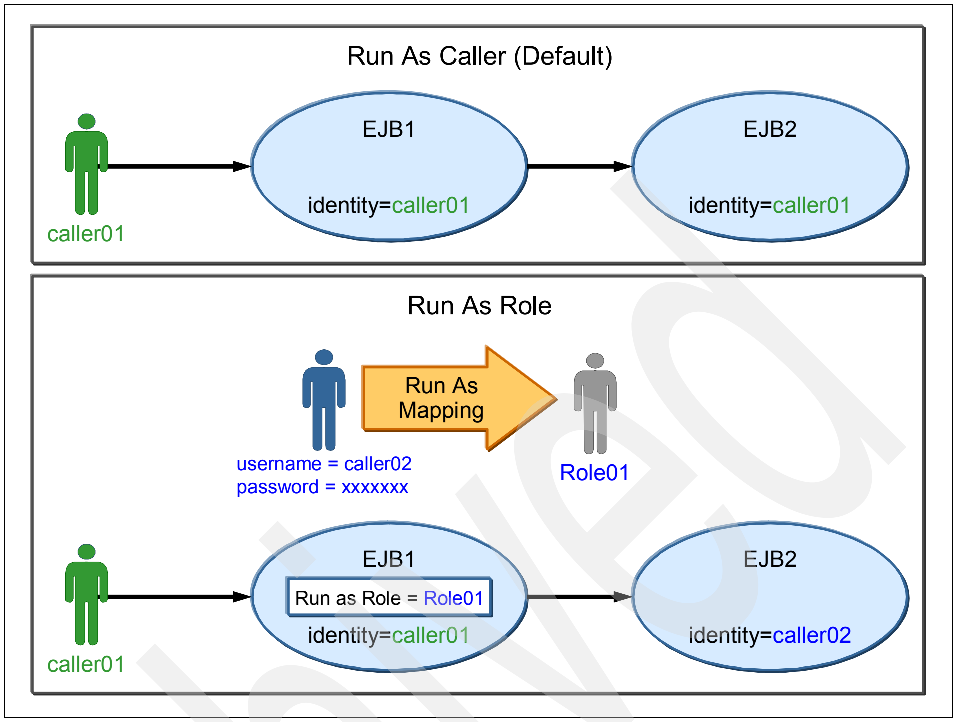


Figure 9-17 Run as Caller compared to Run as Role

Using annotation

The @RunAs security annotation defines the role that will be used for delegation. This annotation can be specified at the class level as shown in Example 9-20. The role name given as the annotation value must have been defined earlier using DeclareRole or RolesAllowed annotation.

Example 9-20 Using RunAs annotation

```
import javax.annotation.security.RunAs;

@DeclareRoles({"runAsRole"})
@RunAs("runAsRole")
@Stateless
public class RunAsCaller implements RunAsCallerLocal {
...
}
```

Using the deployment descriptor

Perform the following steps to define delegation using the EJB deployment descriptor:

1. In the Enterprise Explorer view, double-click the application's EJB module to expand it.
2. Double-click the EJB deployment descriptor to open in the EJB Deployment Descriptor Editor. Select the **Design** tab.
3. Expand **EJB Project *(EJBModuleName)*** → **Enterprise Beans**. Add a new enterprise bean with the name RunAsCallerDescriptor and the EJB class com.ibm.itso.security.ejb.RunAsCallerDescriptor (refer to "Adding enterprise beans" on page 336).
4. Select **RunAsCallerDescriptor**, and click **Add**.
5. Select **Security Identity** as shown in Figure 9-18, and click **OK**.

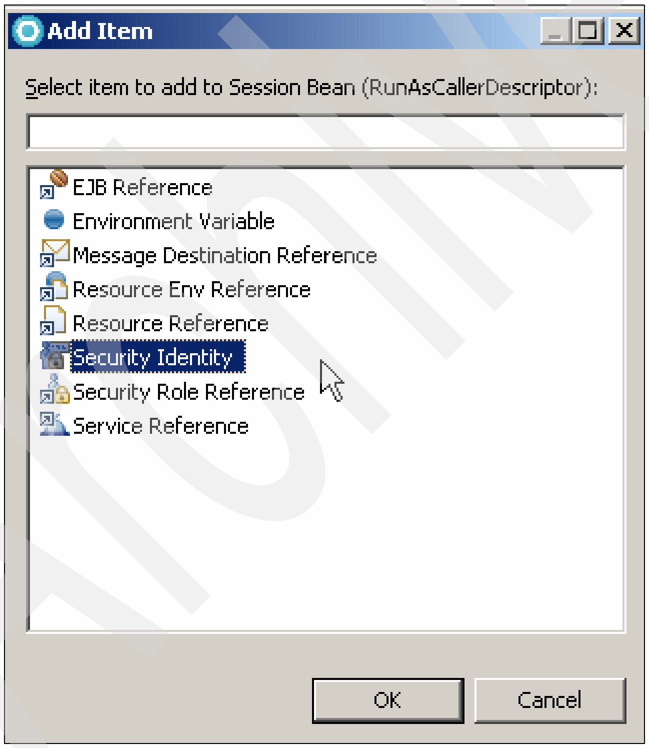
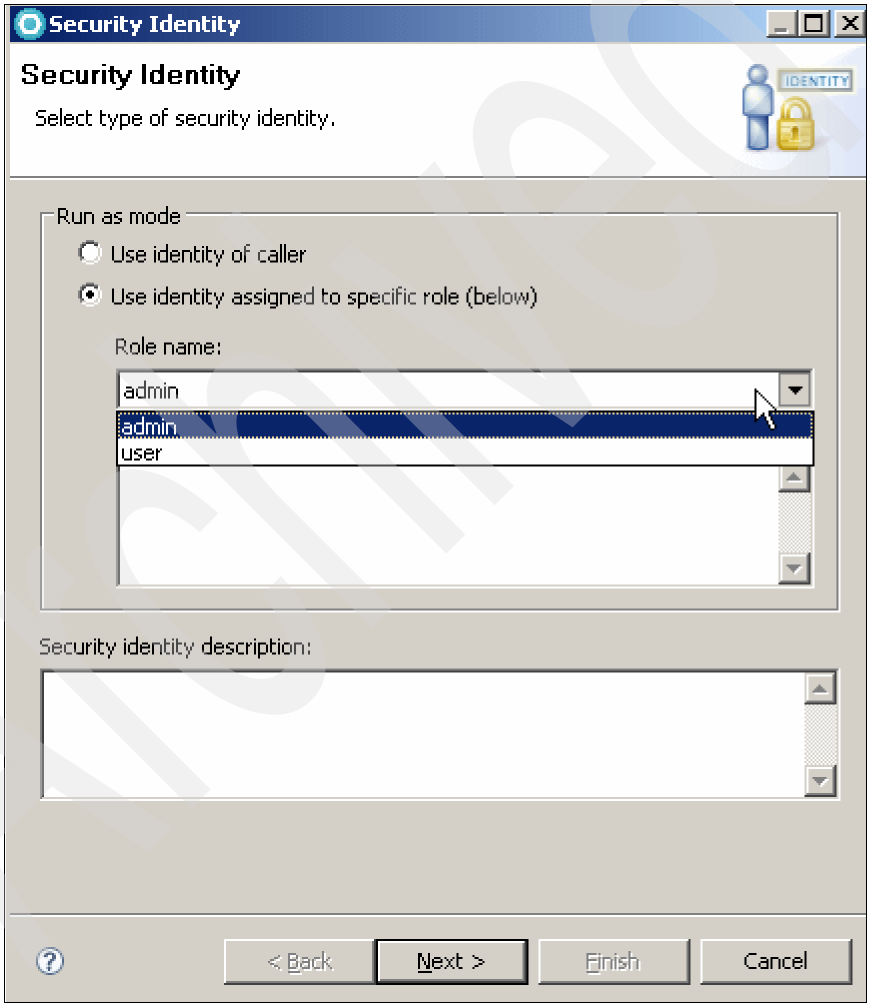


Figure 9-18 Adding Security Identity element

6. The Security identity wizard opens. Select the desired run-as mode:
   - Use identity of caller: This mode is the default setting. The caller identity is used to invoke other beans.
   - Use identity assigned to specific role: The identity mapped to the specified role is used to invoke other beans. You can only select from security roles that have already been defined in the EJB deployment descriptor.

   For this example, select **admin** from the list as shown in Figure 9-19, and click **Next**.



Security Identity

Select type of security identity.

Run as mode

( ) Use identity of caller

(●) Use identity assigned to specific role (below)

Role name: admin

Figure 9-19 Specifying runAs role

7. On the Enterprise Bean Selection panel, select **RunAsCallerDescriptor**, and click **Finish**.
8. Save the EJB deployment descriptor.

A new `<security-identity>` element is added to the RunAsCallerDescriptor bean definition in the EJB deployment descriptor as shown in Example 9-21.

Example 9-21 Security identity element in the EJB deployment descriptor

```
<session>
   <ejb-name>RunAsCallerDescriptor</ejb-name>

<ejb-class>com.ibm.itso.security.ejb.RunAsCallerDescriptor</ejb-class>
   <security-identity>
      <description></description>
      <run-as>
         <description></description>
         <role-name>admin</role-name>
      </run-as>
   </security-identity>
</session>
```

9.6.2 Method-level delegation

WebSphere Application Server supports an extension that allows you to define the RunAs delegation at the method level. With method-level delegation, you can specify various RunAs roles for methods in the same enterprise bean.

Method-level delegation is defined through the EJB extension deployment descriptor (`ibm-ejb-jar-ext.xml`). By default, JEE 5 application does not have deployment descriptors, so you need to generate one. The quickest way is to select the enterprise application project, right-click, and then, select **Java EE** → **Generate WebSphere Extensions Deployment Descriptor** as shown in Figure 9-20.

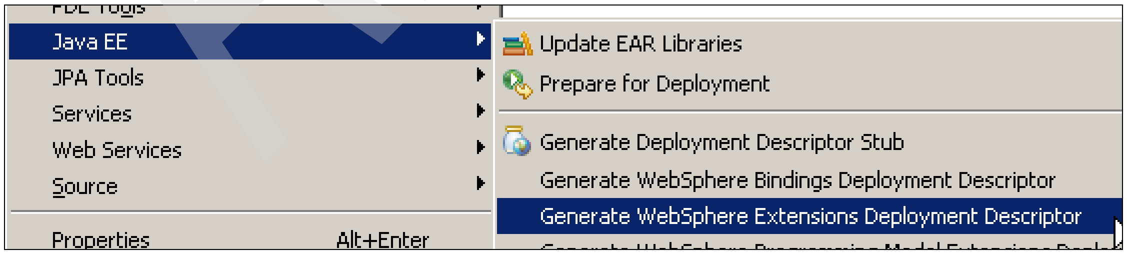

Java EE
JPA Tools
Services
Web Services
Source
Properties Alt+Enter
Update EAR Libraries
Prepare for Deployment
Generate Deployment Descriptor Stub
Generate WebSphere Bindings Deployment Descriptor
Generate WebSphere Extensions Deployment Descriptor

Figure 9-20 Generating EJB extension deployment descriptor

Example 9-22 shows the generated application bindings deployment descriptor.

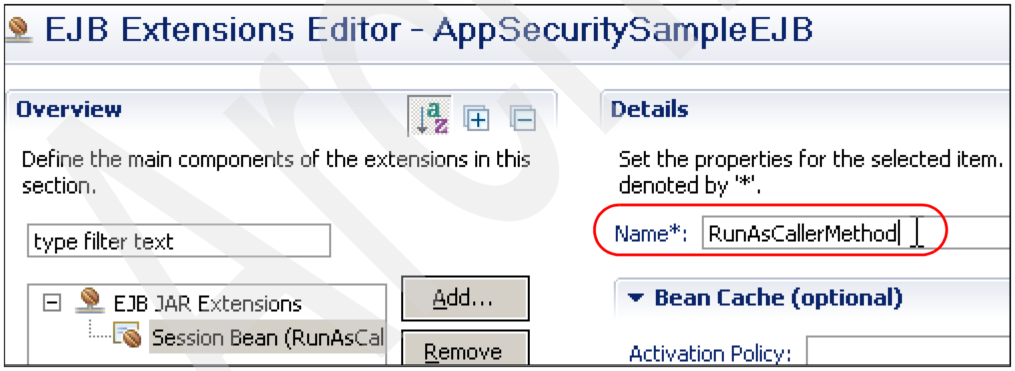
Example 9-22 Generated extensions deployment descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee
http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-ext_1_0.xsd"
   version="1.0">

</ejb-jar-ext>
```

The following example illustrates how to define method-level delegation:

1. In the Enterprise Explorer view, expand the application's EJB module → **ejbModule** → **META-INF**.
2. Double-click **ibm-ejb-jar-ext.xml** to open the EJB Extensions Editor. Select the **Design** tab.
3. Select **EJB JAR Extensions**, and click **Add**.
4. Select **Session Bean**, and click **OK**.
5. Select the newly added Session Bean element, and enter the EJB name as shown in Figure 9-21.


EJB Extensions Editor - AppSecuritySampleEJB
Overview
Define the main components of the extensions in this section.
type filter text
EJB JAR Extensions
Session Bean (RunAsCal
Add...
Remove
Details
Set the properties for the selected item. denoted by '*'.
Bean Cache (optional)
Activation Policy:

Name*: RunAsCallerMethod

Figure 9-21 Specifying bean name

6. Select **Session Bean**, and click **Add**.
7. Select **Run As Mode** as shown in Figure 9-22 on page 359, and click **OK.**

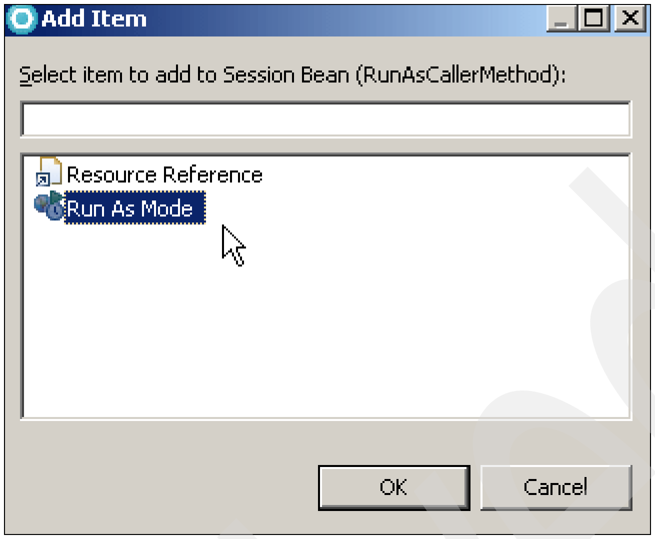


Figure 9-22 Adding Run As Mode element

8. Select the new Run As Mode element and select the desired mode:
   - CALLER_IDENTITY: The caller identity is used to invoke other beans.
   - SPECIFIED IDENTITY: An identity mapped to a specified role is used to invoke other beans.

     When SPECIFIED_IDENTITY is selected as the mode, you need to provide the Role name also.
   - SYSTEM_IDENTITY: The server identity is used to invoke other beans.

> **Note:** The RunAs system identity delegation only works when the server ID and password are used. When the internalServerId feature is used, it does not work, because runAs with system identity is not supported. When internalServerID is used, use RunAsSpecified with a user ID and password that are mapped to the administrator role.
>
> Choosing between internalServerId or specified serverID is done on the user registry configuration page. You can access it by clicking **Security** → **Global Security** and clicking **Configure** in the User account repository section.

For this example, select **CALLER_IDENTITY** as shown in Figure 9-23 on page 360.

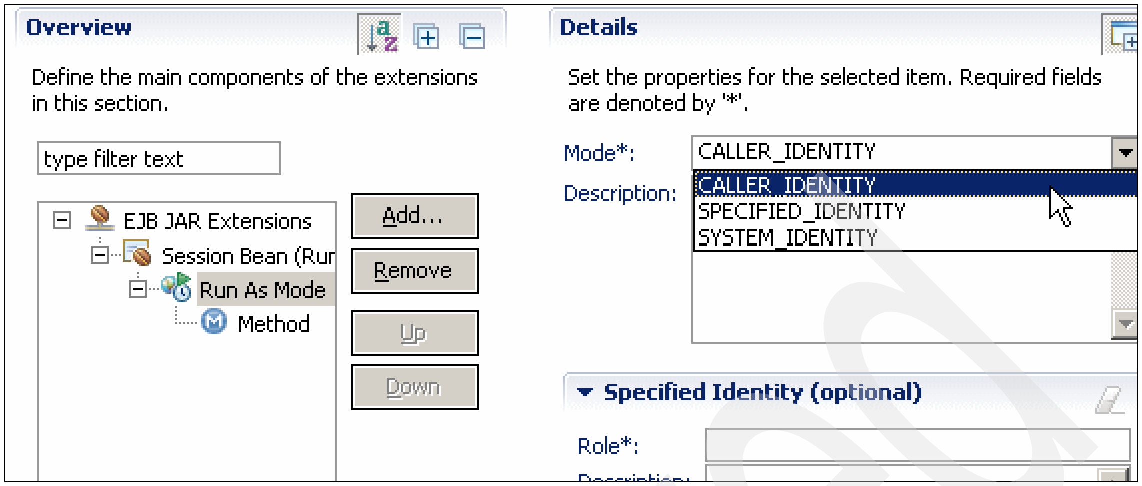


Figure 9-23 Selecting Run As Mode

9. Select **Method** and enter the method name that will use the Run As Mode as shown in Figure 9-24.

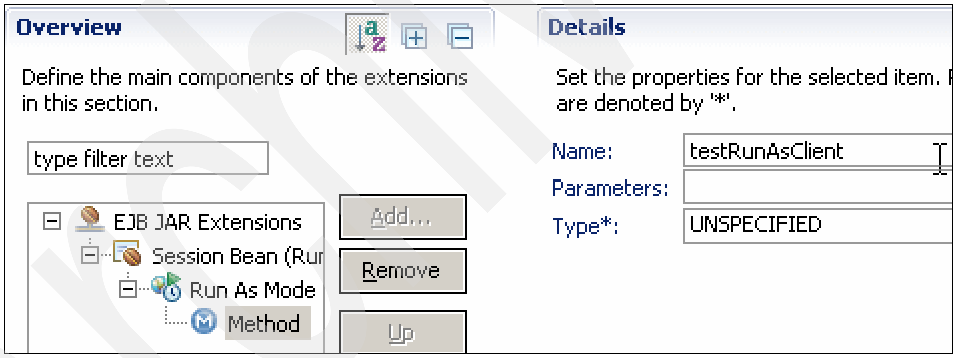


Figure 9-24 Specifying the method name

10. Repeat steps 6-10 and add two additional Run As elements with following parameters:
    - First entry:
        - Mode: **SPECIFIED_IDENTITY**
        - Role: `admin`
        - Method name: `testRunAsRole`
    - Second entry:
        - Mode: SYSTEM_IDENTITY
        - Method name: `testRunAsSystem`

11.Save the extension deployment descriptor.

The new `<run-as-mode>` elements are added to the EJB extension deployment descriptor as shown in Example 9-23.

Example 9-23 Run As Mode elements in the extension deployment descriptor

```
<session name="RunAsCallerMethod">
   <run-as-mode mode="CALLER_IDENTITY">
      <method type="UNSPECIFIED" name="testRunAsClient"/>
      </run-as-mode>
   <run-as-mode mode="SPECIFIED_IDENTITY">
      <specified-identity role="admin"/>
      <method type="UNSPECIFIED" name="testRunAsRole"/>
   </run-as-mode>
   <run-as-mode mode="SYSTEM_IDENTITY">
      <method type="UNSPECIFIED" name="testRunAsSystem"/>
   </run-as-mode>
</session>
```

9.7 Java Authentication and Authorization Service

In rare cases, standard authentication mechanisms might be not sufficient. WebSphere Application Server provides Java Authentication and Authorization Service (JAAS) to allow you to use customized login mechanisms.

More information: A detailed explanation of the JAAS framework and its components is out of the scope for this chapter. You can obtain more information in the WebSphere Application Server Information Center:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/tsec_jaasauthentprog.html`

Example 9-24 on page 362 shows how to use JAAS to perform programmatic authentication to the WebSphere security run time. In this example, a message-driven bean retrieves user data from the message, logs it, and calls a session bean using the new credentials.

Example 9-24 Programmatic login

```
public void onMessage(Message message) {
   // retrieve user data from the message
   ...
   // create login context
   LoginContext lc = new LoginContext( "WSLogin",
      new WSCallbackHandlerImpl(userid, password));
   // do login
   lc.login();
   // retrieve subject
   Subject subject = lc.getSubject();
   // set new identity
   WSSubject.setRunAsSubject(subject);
   // call session bean
   callee.ping();
   // logout - destroy credentials
   lc.logout();
   ....
}
```

In this example, note the following information:

- Example 9-24 uses one of the standard login configurations: WSLogin. This login is a generic JAAS login configuration that can be used by Java clients, client container applications, servlets, JavaServer Pages (JSP) files, and EJB components to perform authentication based on a user ID and password, or a token.
- In addition to the login configuration, a callback handler is also needed to get the authentication data. Because this example is server-side authentication, it has to use the non-prompt handler implementation, which is called *WSCallbackHandlerImpl.*
- The WSSubject.setRunAsSubject method sets a new run as identity on the current thread. The call, `callee.ping()`, uses this new identity.
- At the end, the lc.logout() method is called to clean up and ensure that there are no credential leaks.

9.7.1 WSSubject API

The WSSubject class allows a developer to get information about the current user. This class is available in the WebSphere containers. This list contains the most useful methods in this class:

- String getCallerPrincipal() returns the user name of the current user.
- Subject getCallerSubject() returns the complete subject of the caller.
- doAs() methods methods allow the code to be executed under a different identity. The doAs method requires a separate object for execution. Refer to Example 9-25.

Example 9-25 Using a doAs() method

```
//login programmatically
...
//do something using identity
WSSubject.doAs(subject, newRunner(out));
//logout
...

class Runner implements java.security.PrivilegedAction{
   PrintWriterout;
   publicRunner(PrintWriterout) {
      this.out= out;
   }
   public Object run() {
      doSomethingSecure();
      return null;
   }
};
```

- Subject getRunAsSubject() returns the current runAs subject.
- Void setRunAsSubject() is similar to the doAs method, but it does not require a separate object (Example 9-26).

Example 9-26 Using setRunAs method

```
Subject s = // login here
WSSubject.setRunAsSubject((s);
doSomethingSecure();
// logout here
....
```

Example 9-27 shows how to get additional information about the current user (security name, unique id, realm, groups, and so forth).

Example 9-27 Getting additional information about the current user

```
Subject callerSubject = WSSubject.getCallerSubject();
Set<WSCredential> credentials =
   callerSubject.getPublicCredentials(WSCredential.class);
// should contain only one credential
int credSize = credentials.size();
if( credSize != 1)
   throw new RuntimeException(“Invalid credential number: ”+credSize);
SCredential cred = credentials.iterator().next();
System.out.println("getSecurityName: " + cred.getSecurityName() );
System.out.println("getUniqueSecurityName: " +
   cred.getUniqueSecurityName());
System.out.println("getRealmName: " + cred.getRealmName());
System.out.println("getRealmSecurityName: " +
   cred.getRealmSecurityName());
System.out.println("getRealmUniqueSecurityName: " +
   cred.getRealmUniqueSecurityName());
// always return null
System.out.println("getRoles: " + cred.getRoles());
ArrayList groupIds = cred.getGroupIds();
System.out.println("getGroupIds: " + groupIds);

// Sample output from the above code:
getSecurityName: gas
getUniqueSecurityName: uid=gas,o=defaultWIMFileBasedRealm
getRealmName: defaultWIMFileBasedRealm
getRealmSecurityName: defaultWIMFileBasedRealm/gas
getRealmUniqueSecurityName:
defaultWIMFileBasedRealm/uid=gas,o=defaultWIMFileBasedRealm
getRoles: []
getGroupIds:
[group:defaultWIMFileBasedRealm/cn=admins,o=defaultWIMFileBasedRealm]
```

Note: Currently, you cannot get information about user roles. The WSCredential.getRoles() method always returns null. It is provided for future enhancements.

9.8 Using annotations

The following rules apply to using annotations:

- The PermitAll, DenyAll, and RolesAllowed annotations cannot be applied on the same method or class.
- The method-level annotations take precedence over the class-level annotation.
- Settings in a deployment descriptor take precedence over annotations.
- Annotations are not inherited, yet they can apply to methods, which are inherited. So if a method is overridden, the annotation is overridden (the super class annotation is lost). This override is explained in Example 9-28.

Example 9-28 Annotation inheritance

```
@RolesAllowed("user")
public class ParentClass {
   public void aMethod() { ... }
   public void bMethod() { ... }
   public void cMethod() { ... }
}

@Stateless
public class InheritanceTester extends ParentClass implements
InheritanceTesterLocal {
   // overrides annotation explicitly
   @RolesAllowed(value={"admin"})
   public void aMethod() { ... }

   // overrides method - uses current class role assignments
   public void bMethod() { ... }

   // cMethod() not overridden - uses parent class role assignments
}

Assuming that aMethod, bMethod and cMethod are part of the
InheritanceTesterLocal interface, method permitions in the
InheritanceTester bean are as follows:

– aMethod - @RolesAllowed(“admin”)

– bMethod - not specified

– cMethod - @RolesAllowed(“user”)
```

Best practice: Although it is legal to use annotations and EJB deployment descriptor entries at the same time to specify security settings, to avoid confusion, we strongly recommend that you use either annotations or the EJB deployment descriptor.

A comparison of annotations and deployment descriptor settings is shown in Table 9-1.

Table 9-1 Annotations compared to deployment descriptors

Security annotation	Deployment descriptor
Security settings are defined in the Java class file but they can be overridden by descriptor settings.	The security settings are in an external XML file.
Annotations are defined by the Java developer, also known as the *Bean Provider*.	The deployment descriptor is usually defined by the Java developer, but it can also be defined later by Assembler.
Annotations are easier to define during development; just add the annotation in the class.	The deployment descriptor is harder to define during development; you need to create and maintain the descriptor.
Annotations are harder to change; you need to change and recompile Java source files.	The deployment descriptor is easier to change; just change the XML file without touching the application code.
Annotation syntax knowledge is required.	XML descriptor syntax knowledge is required.
It is harder to see the entire application security view. Annotations are scattered among many classes. There are currently no tools in Rational Application Developer to see a generated descriptor easily.	It is easier to see the entire application security view, because everything is defined in one descriptor.
Annotations still require separate descriptors to define bindings and extensions.	Bindings and extensions are in separate descriptors.
Annotations are useless for web modules. Most of the information must be in the deployment descriptor (login configuration, security constraints, and so forth). Only the security roles and the runAs role can be defined using annotations.	The deployment descriptor must be used for Web modules anyway.

For more information, refer to these resources:

- *Common Annotations for the Java Platform at:*

  `http://jcp.org/aboutJava/communityprocess/final/jsr250/index.html`
- Enterprise JavaBeans, Version 3.0 core contracts and requirements at:

  `http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html`

Part 3

z/OS specifics

10

WebSphere z/OS and local operating system security

Protecting sensitive and confidential data from malicious intruders, deadly viruses, and worms is not an easy task. It requires constant monitoring of the daily IT business operations and deploying the latest security technology. WebSphere Application Server for z/OS security and the underlying operating system infrastructure security can provide clients running enterprise applications with secure and reliable services. These services are consistently managed by a security product, such as IBM Resource Access Control Facility (RACF). This chapter will focus on the use of RACF as the underlying infrastructure for WebSphere Application Server for z/OS security.

This chapter contains the following sections:

10.1 Local operating system (LocalOS) security

The security infrastructure of the underlying operating system provides certain services to the WebSphere security application. This service includes the file system security support to secure sensitive files in WebSphere product installation. The WebSphere system administrator can configure the product to obtain authentication information directly from the operating system user registry. On z/OS, the WebSphere LocalOS registry option leverages System Authorization Facility (SAF). SAF is a common set of APIs supporting the ability for a pluggable z/OS security manager, such as RACF or a third-party equivalent, to manage the authentication and authorization needs on z/OS. These security products contain information about users, groups, resources, and resource access permissions. The purpose of these products is to provide authentication and access control for the z/OS environment.

RACF provides the following capabilities and functionality:

- Create and manage digital certificates.
- Protect data sets and UNIX System Services hierarchical file system (HFS) files.
- Protect system resources and services.
- Manage a large user database, and add, delete, list, and change user profiles.

Authentication and authorization in RACF is very straightforward. When a user requests a service from the system, RACF first checks whether the user is defined to RACF. If yes, RACF checks whether the user is authorized to access that resource. A user can be in a suspended or revoked state and is not given any system privilege until the suspension or revocation is resolved.

RACF keeps a profile for each system resource user that it knows, and the profile is kept in storage in the format that is shown in Figure 10-1.

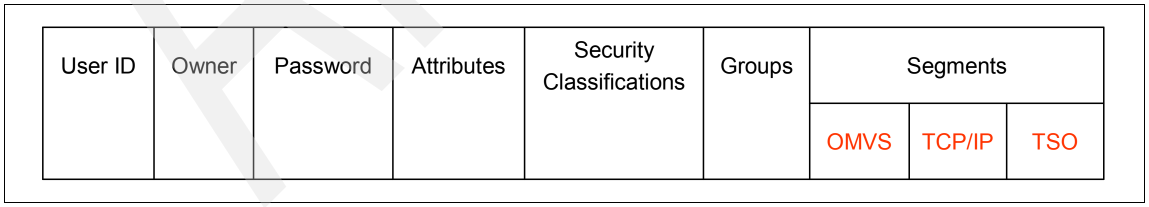


Figure 10-1 RACF profile

As you can see in Figure 10-1 on page 372, the RACF profile contains:

- *User ID* and *password*. The password is encrypted.
- *Owner* of the profile. The owner can be a group or a single user.
- *Attributes*, which allow users to perform specific tasks with RACF. Five attributes that can be specified are *special*, *auditor*, *operations, restricted*, and *protected.*
- *Security classifications* are optional, but they give an additional way of controlling user authority.
- *Segments* are z/OS process or address spaces, such as TSO, OMVS, and CICS® , that can be added to a user profile.

Finally, RACF provides a way to record what is done on the system. It keeps track of what happens on the system so that an organization can monitor who is logged on the system at any time. RACF reports whether anyone has attempted to perform unauthorized actions. For example, RACF can record that a user has tried to access or change data to which that user does not have the correct authority.

Access Control Element

The *access control environment element (ACEE)* is storage (a control block) that represents a user or task credentials, which contain the profile information of a user who is currently active. ACEE is constructed and anchored on the z/OS address space, representing the credentials for the address space. Optionally, another ACEE can be anchored on the Task Control Block (TCB), representing the credential of the executing thread.

10.2 User ID strategy for a Network Deployment environment

When planning for the customization of a cell, it is expedient to think about the user ID strategy for the complete Network Deployment environment. During the customization, the user IDs of all necessary started tasks (STC) is defined. As a result, the RACF jobs will be generated to create the IDs. Changing user IDs in an existing environment can be very painful, because they have an impact on all RACF profiles, certificates, and keyrings. In practice, the fastest way to get all required definitions for the new user ID is to regenerate the RACF jobs for each node using Profile Management Tool (PMT).

As illustrated in Figure 10-2 on page 374, strategies for assigning user IDs range from a *common* approach, where all control regions share one user ID and all servant regions share another user ID, to a *unique* approach, where each address space has a unique user ID.

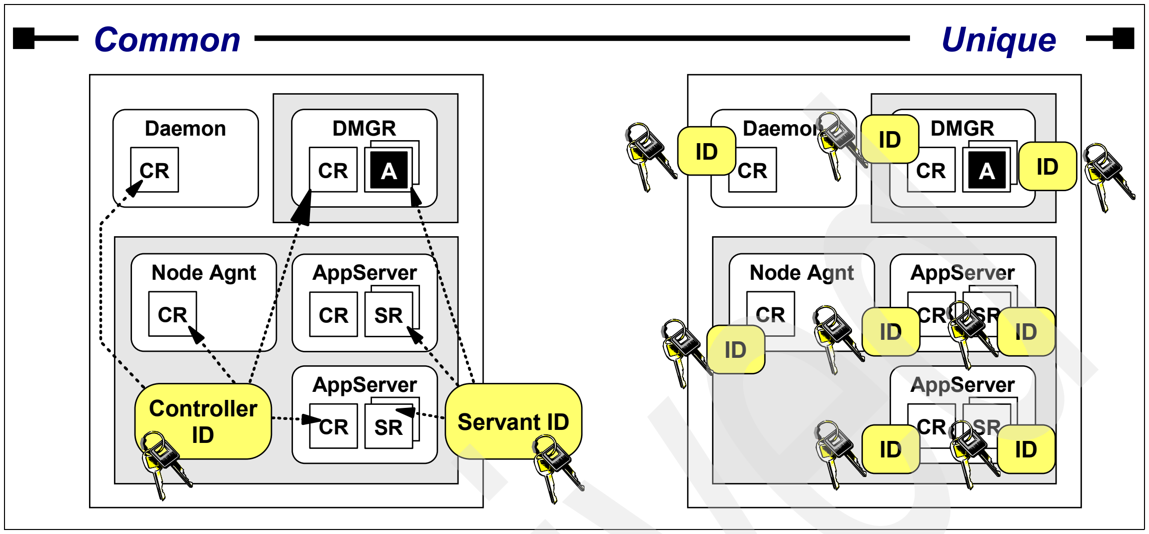


Figure 10-2 Common and unique started task user IDs

Many clients have implemented the common approach. It is simpler and requires fewer SAF profile definitions, particularly when adding new nodes and servers. For the common approach, the number of SAF profiles can be dramatically reduced by implementing generic RACF profiles as pointed out in 10.6.1, “Generic RACF profiles using wildcards” on page 401.

We recommend that you at least differentiate between the control region and the servant region as outlined in the common approach, because it isolates the IBM-authorized code that runs in the control region from the application code that runs in the servant region. The disadvantage of the common approach is that it becomes more difficult to find the origin of a security violation.

In contrast, the unique approach offers the highest level of isolation and the ability to perform charge backs in an organization where WebSphere environments are departmentalized. The disadvantage of this solution is that a new SAF keyring is required for every new user ID. For an insolated environment, sharing the certificates across the cell is not an option. Consequently, new certificates need to be generated for each keyring. Moreover for each new application server, many SAF profiles need to be defined. As a result, the security administration becomes more complicated.

The required RACF profiles for a new server with a unique user ID are explained in the WebSphere Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/csec_safclasses.html

In practice, a user ID strategy will be selected somewhere between both approaches. A good approach is, for instance, to follow the common approach as long as an application does not have special requirements regarding security. Applications with special security requirements can then be isolated using separate user IDs for that dedicated application server.

In addition, a user ID strategy based more on the common approach can take advantage of the generic RACF profiles that are described in 10.6.1, “Generic RACF profiles using wildcards” on page 401.

Figure 10-3 illustrates an environment that shares user IDs across a horizontal cluster. In this case, the application server *control regions* (CRs) and *servant regions* (SRs) have been isolated from the deployment manager CRs and SR. In addition, the daemon also runs under another user ID.

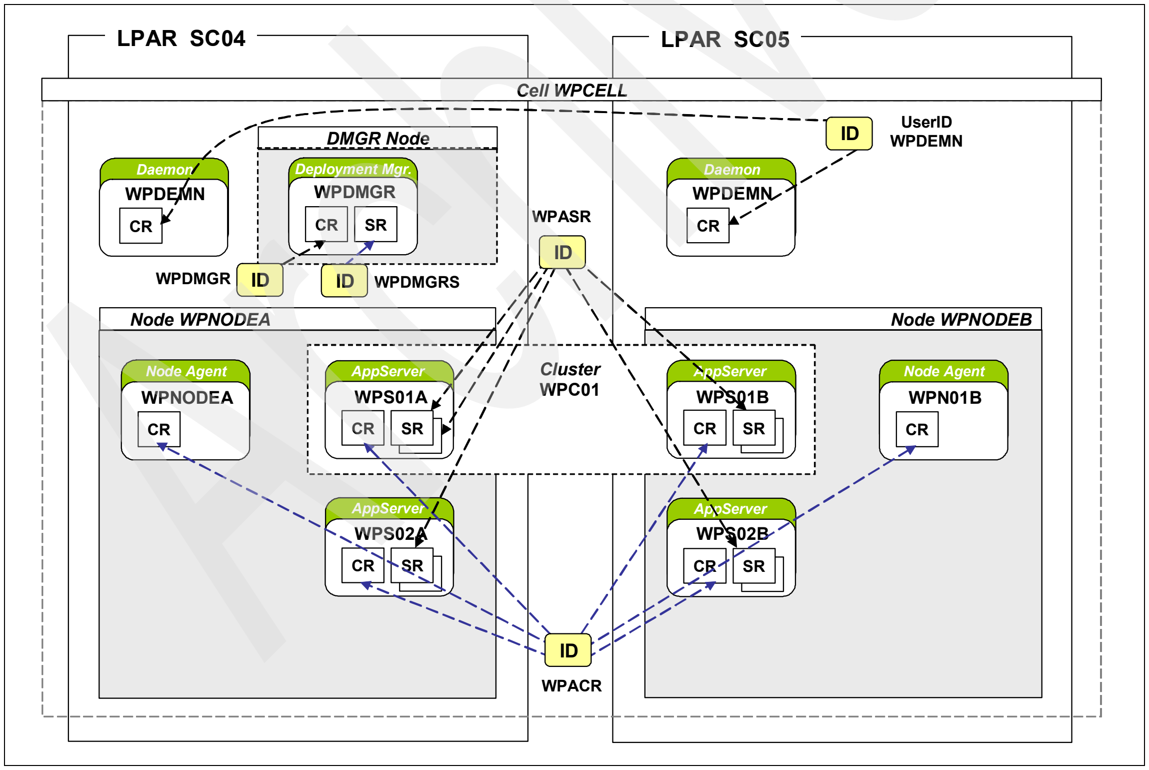


Figure 10-3 Example of a user ID strategy in a cluster environment

10.3 Administrative security

Beginning with WebSphere Application Server V6.1 for z/OS, global security has been split into administrative and application security. *Administrative security* can be enabled during profile creation for a deployment manager, stand-alone application server, custom node, job manager, and the secured proxy. Administrative security, which is already operational, prevents unauthorized access to the administrative console and enables secure connections between each component using Secure Sockets Layer (SSL). In contrast, *application security* must be enabled after profile creation to restrict access to the applications. When application security is activated, the authentication method and the required Java Platform, Enterprise Edition (EE) roles in the deployment descriptor of an application take effect.

With WebSphere Application Server Version 7.0, the deprecated ISPF (Interactive System Productivity Facility)-Panels are no longer available. The new WebSphere Configuration Tool (WCT) with the built-in Profile Management Tool (PMT) have replaced the ISPF-Panel for customizing a WebSphere z/OS environment. The panels that are shown in this section illustrate the security setup options of the PMT dialog.

10.3.1 Common groups and user IDs

During profile customization, you define common *groups* and *users* for the new WebSphere environment according to the selected user ID strategy.

Figure 10-4 on page 378 shows the panel where the following SAF user IDs are defined:

- Administrator user ID

  The server administration user ID needs to be defined with a password. The expiry policies for passwords need to be checked with your security administrators. We advise that you either assign a non-expiring password or have good password reset policies.

  ```
  ALTUSER WTADMIN PASSWORD(password) NOEXPIRED
  ```

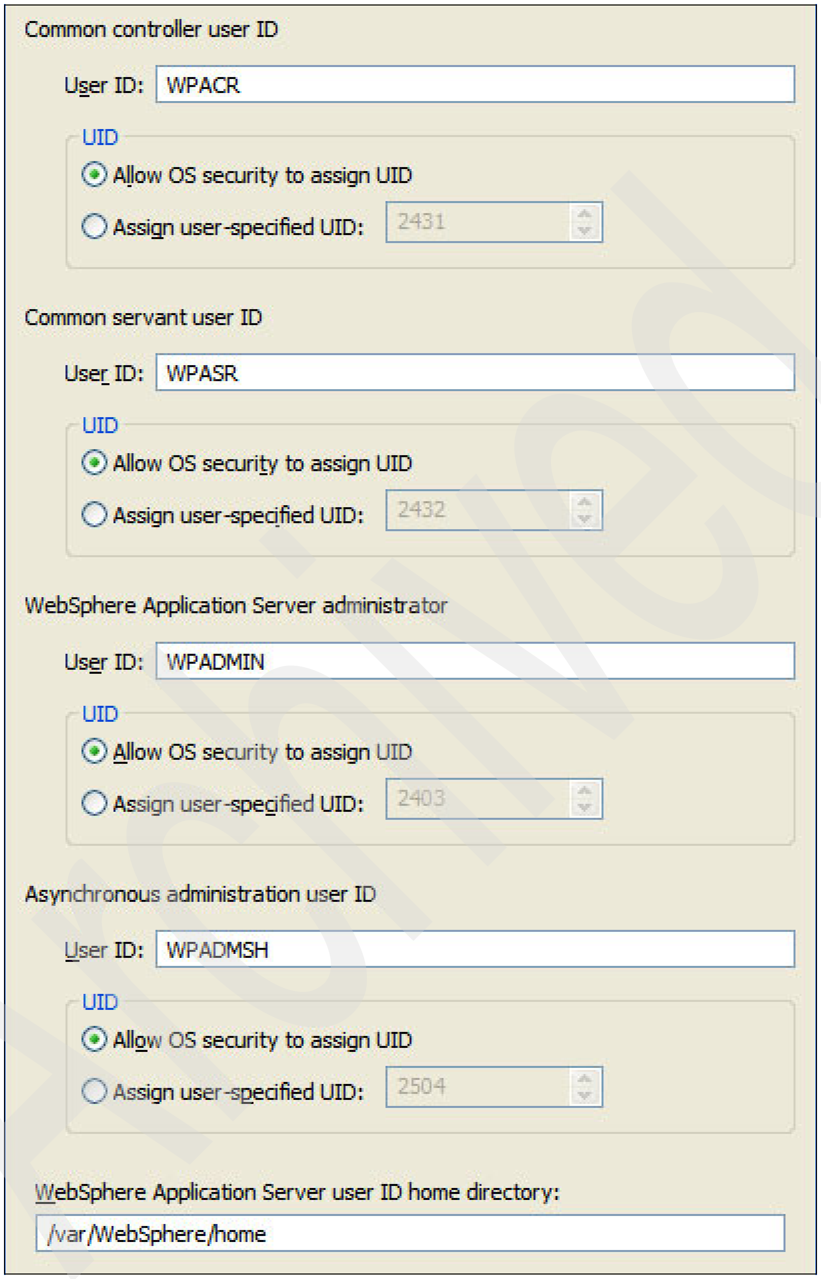
  With Version 6.1, a new user ID was introduced called the *HFS owner* (file system owner), and it disappeared in Version 7.0. The HFS file system is owned by the WPADMIN user, which belongs to the WPCFG group.

- Control region (CR) user ID

  This user ID can be reused for the CRs of additional application servers across the cell.

- Server servant region (SR) user ID

  This user ID can be reused for the SRs of additional application servers across the cell.



Common controller user ID

User ID: WPACR

UID

[✓ Allow OS security to assign UID]

Assign user-specified UID: 2431

Common servant user ID

User ID: WPASR

UID

[✓ Allow OS security to assign UID]

Assign user-specified UID: 2432

WebSphere Application Server administrator

User ID: WPADMIN

UID

[✓ Allow OS security to assign UID]

Assign user-specified UID: 2403

Asynchronous administration user ID

User ID: WPADMSH

UID

[✓ Allow OS security to assign UID]

Assign user-specified UID: 2504

WebSphere Application Server user ID home directory:

/var/WebSphere/home

Figure 10-4 Configure common users

Moreover, the following SAF groups are created during the customization of the WebSphere environment (Figure 10-5 on page 380):

- Configuration group

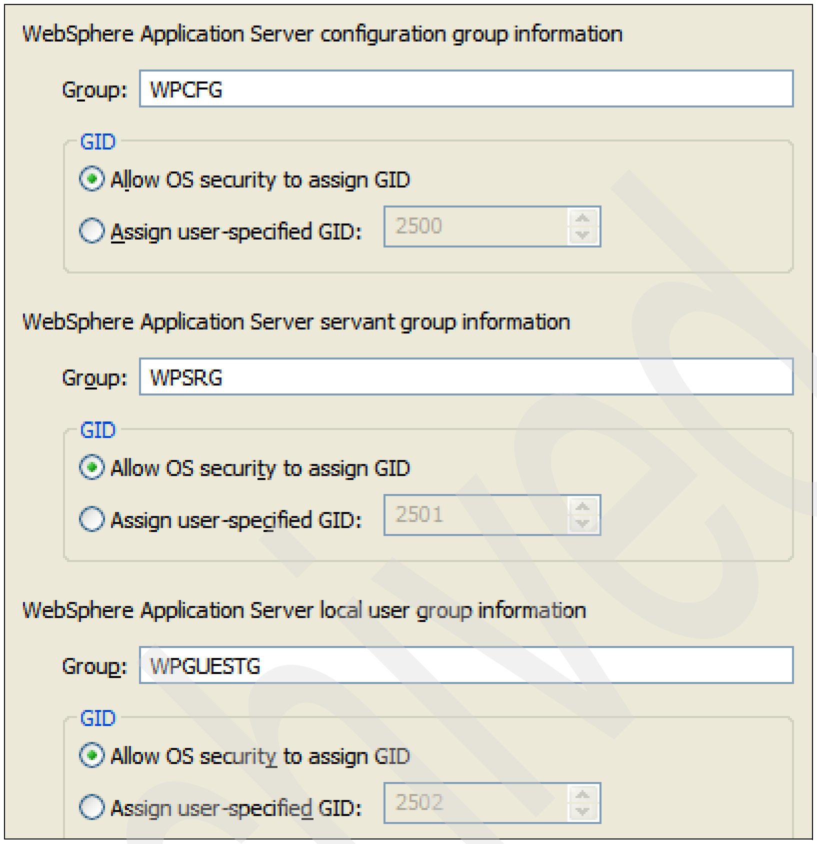
  The *configuration group* contains all started task control (STC) user IDs, such as the daemon, deployment manager CR and SR, application server CR and SR, and the node agent. All permissions in the WebSphere configuration file system are based on the administration user ID and the configuration group. Most of the files in the configuration file system have the permissions 775 and 770, which allow all members of this group to access the configuration file system.

- Servant region group

  All servant region user IDs must be connected to this group. The major advantage of this group is that permissions for connections to other subsystems, such as CICS and DB2, can be granted centrally.

- Local user group

  All unauthenticated user IDs, such as the guest user, belong to this group.



WebSphere Application Server configuration group information

Group: WPCFG

GID

[✓ Allow OS security to assign GID]

Assign user-specified GID: 2500

WebSphere Application Server servant group information

Group: WPSRG

GID

[✓ Allow OS security to assign GID]

Assign user-specified GID: 2501

WebSphere Application Server local user group information

Group: WPGUESTG

GID

[✓ Allow OS security to assign GID]

Assign user-specified GID: 2502

Figure 10-5 Configure common groups

The relationship of the user IDs to groups is shown in Figure 10-6 on page 381.

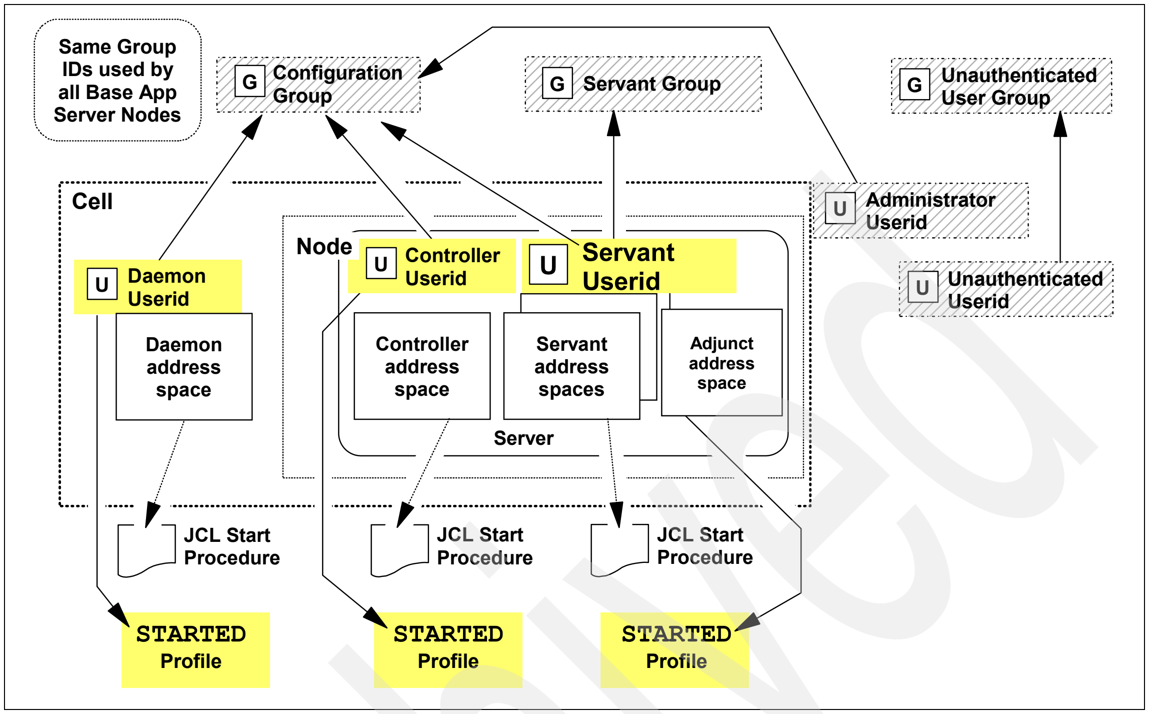
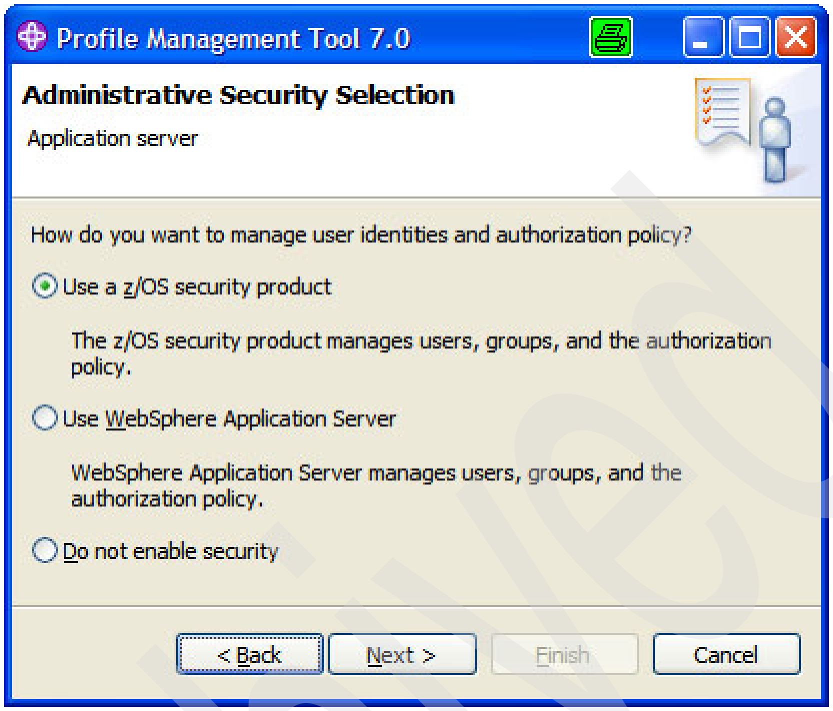


Figure 10-6 Relationship of user IDs to certain groups

10.3.2 Security configuration options

In the customization process, there are three options for the user registry from which you can select as illustrated in Figure 10-7 on page 382.


Profile Management Tool 7.0

Administrative Security Selection

Application server

How do you want to manage user identities and authorization policy?

(●) Use a z/OS security product

The z/OS security product manages users, groups, and the authorization policy.

( ) Use WebSphere Application Server

WebSphere Application Server manages users, groups, and the authorization policy.

( ) Do not enable security

Figure 10-7 Administrative Security: Use a z/OS security product

In the following section, we explain the features of each option.

Option 1: Use a z/OS security product

If you want to enable administrative security and use a local z/OS repository for authentication and authorization, you must choose this option. These characteristics define this option:

- Each WebSphere Application Server user and group identity correspond to a user ID or group in the z/OS system's SAF-compliant security server, such as RACF.
- Access to WebSphere Application Server Java Platform, Enterprise Edition (Java EE) roles is controlled using the SAF EJBROLE class profiles in conjunction with the GEJBROLE class, if you choose to use the grouping class.
- Digital certificates for SSL communication are stored in the z/OS security product.

For WebSphere Application Server for z/OS, SAF is always used to control started task identities and the digital certificate of the location service daemon.

However, if this option is selected, all WebSphere Application Server administrators and administrative groups must be defined to SAF as well.

This option is appropriate when servers or cells reside entirely on z/OS systems, with SAF as the user registry. If you plan to implement a Lightweight Directory Access Protocol (LDAP) or a custom user registry and plan to map LDAP/CUR identities to SAF identities and use SAF authorization, choose this option. That way, the initial SAF EJBROLE setup is performed. We provide more information about LDAP as a user registry in Chapter 12, "WebSphere z/OS and user registries" on page 433.

Option 2: Use WebSphere Application Server security

In option 1, the user authentication and authorization policy is managed through SAF. In all options, SAF is used to control WebSphere Application Server for z/OS started task identities, as well as the location service daemon's digital certificate. However with option 2, users and groups are defined in the WebSphere Application Server user registry, and authorization is managed by WebSphere Application Server. These characteristics define option 2:

- Each WebSphere Application Server user and group identity corresponds to an entry in a WebSphere Application Server user registry. The initial user registry is a *federated repository*, a simple file-based user registry created during customization and residing in the configuration file system.
- Access to WebSphere Application Server roles is controlled using WebSphere role bindings. In particular, administrative role bindings are set through the Administrative User Roles and Administrative Group Roles settings in the administrative console.
- Digital certificates for SSL communication are stored in the configuration file system. All necessary server certificates are created in the configuration file system automatically by WebSphere Application Server.

When option 2 is selected, an additional panel is provided for you to specify the administrator user ID. This user ID is stored in the file-based user registry and used as the initial administrative user. The sample user ID is optional. This field appears after you have selected the option to install the sample application.

Option 3: Do not enable security

If you select **Do not enable security**, administrative security will not be activated.

Recommendation

We recommend that you enable global security during the customization process of the cell and that you select **Use a z/OS security product**.

As a result of this selection, PMT will generate a RACF job, including all necessary RACF profiles, certificates, and keyrings for an secured environment. If “Do not enable security” or “Use WebSphere Application Server” is selected, these additional RACF definitions will not be generated. Moreover, the keystores and truststores, including the certificate, will be located in the UNIX file system of the WebSphere configuration and not in SAF. If you decide to use the SAF authorization provider in a later stage, these RACF definitions need to be defined manually. For SSL configuration, the keystore, as well as the truststores, must be regenerated in the administrative console to use SAF-managed keyrings.

We also recommend that you use a z/OS security product for the following reasons:

- Risk of falsification of data

  RACF user control is much more difficult to be falsified than WebSphere Application Server security user control, in which case information is stored in flat files.

- Easy to change security configuration

  Regarding the RACF setup, it is easier to change from z/OS security to another option than from another option to z/OS security.

In development environments, there are sometimes requirements to disable global security. In this case, it is a common approach to enable global security with the SAF authorization provider during the customization process and turn off administrative security after cell construction. The benefit of this solution is the flexibility to turn on administrative security without the need to define additional SAF profiles, certificates, and keyrings.

In certain cases, global security has been disabled during the construction of a Network Deployment environment in order to avoid SSL problems with the federation of nodes. These problems are mainly caused by untrusted CA certificates, which must be added to the keyring of the job submitter. There is no need to turn off security during a node federation.

Comparison of options

For a comparison of WebSphere Application Server security configurations, refer to Table 10-1.

Table 10-1 Comparison of default security settings

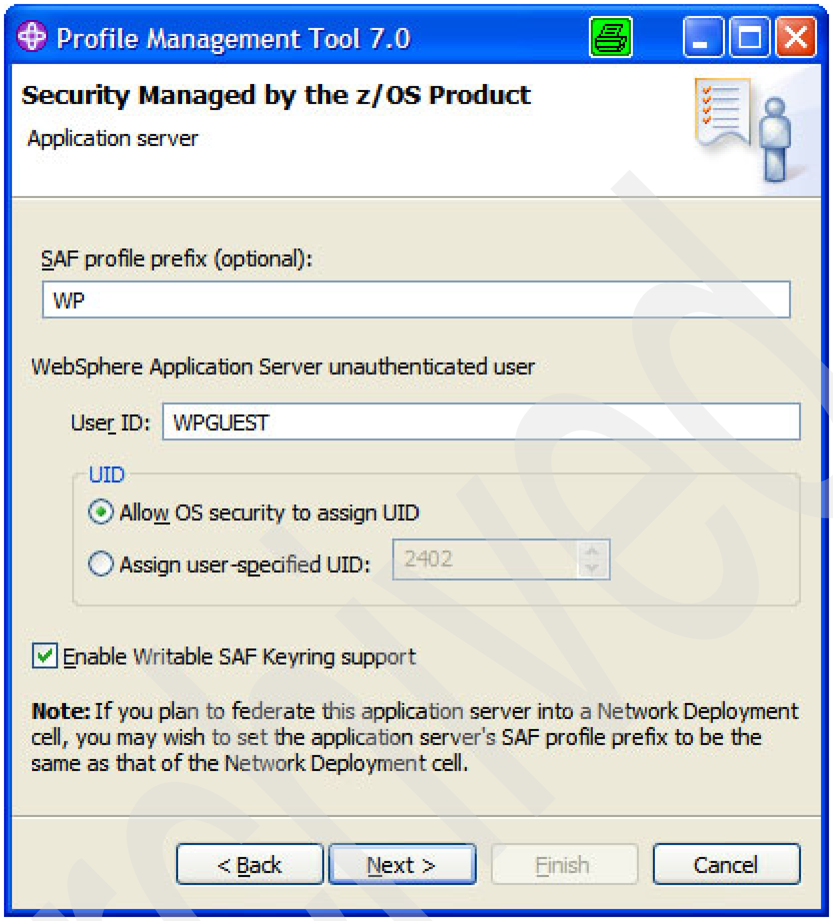
	z/OS security	**WebSphere Application Server security**
User account repository	Local operating system: User IDs, groups, and passwords are stored in SAF.	Federated repository: User IDs and passwords are stored in a file-based placed UNIX System Service. User IDs and passwords are stored in `fileRegistry.xml`.
Authorization technology (external authenticated provider security)	System Authorization Facility (SAF) authorization: Uses the authorization policy that is stored in SAF.	Built-in authorization: Uses WebSphere Security authorization.
Administrative role definition	SAF EJBROLE profile: Mapping information is stored in SAF. Definition of EJBROLE profiles is required. (Default roles are defined during server installation process.)	WebSphere Admin Role: Mapping information is stored in UNIX System Services: ► Information of user IDs and roles is stored in `admin-auth.xml`. ► Primary admin user ID, which is used to log on to the administrative console, is stored in `security.xml`.
SSL configuration: Key store type	System Authorization Facility keyring: Stored in RACF DB Key store PATH: safkeyring:///*<keyring_name>* Type: JCERACFKS	File-based key store: Stored in UNIX System Services. Key store PATH: ${CONFIG_ROOT}/cells/*<cell_name>*/nodes/*<node_name>*/*<key_file>* Type: PKCS12

	z/OS security	**WebSphere Application Server security**
Application deployment (role mapping)	Security Authorization Facility (SAF)-based mapping: ► Need to define new EJBROLE profiles in RACF. ► The “Map security roles to users or groups” step in the deployment of an application is ineffective.	File-based mapping: Need to map user and role in the “Map security roles to users or groups” step in the deployment of application.
BBOxBRAK (additional RACF commands)	Certificates Keyrings CBIND EJBROLE Sync-to-thread EnableTrustedApps	None

Note: The following topics assume that the z/OS security product has been chosen. Consequently, the z/OS security product will be the user registry (LocalOS), as well as the authorization provider.

10.3.3 z/OS security product options

When you select the z/OS security product options, you have additional configuration options as shown in Figure 10-8 on page 387.



Profile Management Tool 7.0

Security Managed by the z/OS Product

Application server

SAF profile prefix (optional): WP

WebSphere Application Server unauthenticated user

User ID: WPGUEST

UID

(●) Allow OS security to assign UID

( ) Assign user-specified UID: 2402

[✓] Enable Writable SAF Keyring support

Note: If you plan to federate this application server into a Network Deployment cell, you may wish to set the application server's SAF profile prefix to be the same as that of the Network Deployment cell.

Figure 10-8 Security Managed by the z/OS Product

SAF profile prefix

The *SAF profile prefix*, formerly known as the SAF security domain, specifies a <*domain*> prefix, which will be prepended to all CBIND and EJBROLE profiles. If your environment consists of multiple cells within one logical partition (LPAR) or sysplex, this prefix can be used to distinguish the administrator role in each cell. A common approach is to use the two-character abbreviation of the cell (in this case, `WP`) as the SAF profile prefix.

If the SAF profile prefix is not specified, the corresponding RACF for a CBIND profile is structured this way:

```
RDEFINE CBIND CB.BIND.** UACC(READ)
PERMIT CB.BIND.** CLASS(CBIND) ID(WPCFG) ACCESS(CONTROL)
```

If he SAF profile prefix identifier is specified, the domain name is added to the profile. The resulting RACF command is:

```
RDEFINE CBIND CB.BIND.<domain>.** UACC(READ)PERMIT CB.BIND.<domain>.**
CLASS(CBIND) ID(WPCFG) ACCESS(CONTROL)
```

The following WebSphere Information Center article provides a good overview of all SAF prefix cases:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/csec_safsecdom.html

In 10.5.1, "SAF authorization for administrative roles" on page 393, the EJBROLE profiles, including the `<domain>` prefix, are explained in detail.

Unauthenticated user ID

All users within z/OS need an identity. In particular, if sync-to-thread is enabled and the user has not authenticated so far, an identity needs to be passed to the operating system. In this case, the unauthenticated user ID is used. In WebSphere z/OS, all unauthenticated users are represented by this user ID. It is common practice to define the unauthenticated user ID with both the *restricted* and *protected* attributes.

Enable writable SAF keyring support

The *writable keyring support* enhances the capabilities of the administrative console to consistently manage keyrings and certificates, which are stored in SAF. If this option is enabled, the required RDATALIB profiles are generated in the RACF job for the writable keyring support. We recommend that you use the writable keyring support for the increased efficiency of managing SAF keyrings and certificates. We provide more information regarding the writable keyring support in 13.4, "Writable SAF keyrings" on page 493.

10.4 RACF jobs for WebSphere z/OS

This section discusses the RACF definitions and the jobs that are created to define them.

10.4.1 RACF classes

This section describes the pertinent RACF classes that are applicable to implementing WebSphere Application Server for Version 7.0 on z/OS. These classes represent a small subset of all available classes in RACF:

- APPL: The APPL class controls access to applications, including the administrative console.
- CBIND: The CBIND class controls the client's ability to bind to the server. WebSphere uses the CBIND class to control access to the server.
- DIGTCERT: The DIGTCERT class contains digital certificates and related information.
- DSNR: The DSNR class controls access to DB2 subsystems.
- EJBROLE and GEJBROLE: These classes are used to register Enterprise JavaBeans (EJB) roles that will be used by WebSphere Application Server applications. EJBROLE is the member class for EJB authorization roles. The APPLDATA field in an EJBROLE profile defines the target Java identity when running in RunAs ROLE mode. GEJBROLE is the grouping class for EJB authorization roles. EJBROLE profiles have to be added for the required roles and for users to be given access to these profiles when SAF authorization is used.
- FACILITY: This class is for miscellaneous uses. Profiles are defined in this class so that resource managers can check users' access to the profiles when the users take an action. We place profiles for Digital Certificate, Distributed Computing Environment (DCE), and Kerberos, plus UNIX System Services profiles (for example, BPX.DAEMON), in this class.
- SERVER: This class controls the server's ability to register with the daemon. This class is used in WebSphere to control whether a servant can call authorized programs in the controller.
- SERVAUTH: The SERVAUTH class contains profiles that are used by servers to check a client's authorization to use the server or to use the resources managed by the server. Use this class to protect TCP/IP ports. If you use this class, you must give WebSphere and Kerberos access.
- STARTED: This class is used for identifying authorized system started procedures. WebSphere Application Server normally starts as a system task and needs an entry in the STARTED class to associate a valid RACF user ID and connected group to be able to access protected resources. This class is used in preference to the started procedures table to assign an identity during the processing of an MVS START command. For example, WebSphere and Kerberos are defined as started tasks in this profile.

- SURROGAT: This class determines whether surrogate submission or login is allowed, and if allowed, which user IDs can act as surrogates. SURROGAT is used here in conjunction with BPX.SRV profiles in the SURROGAT class to allow security context switches for unauthenticated user IDs.
- RDATALIB: This class is used for control access to SAF keyrings and certificates.

10.4.2 Skeleton of the generated RACF jobs

Regardless of the administrative security option selected, you submit the customization jobs to define the desired security infrastructure as part of the server configuration process. Profiles in the STARTED class, for instance, need to be defined in any case when running WebSphere on z/OS. In the process of configuring a base application server, member *hlq*.DATA(BBOWBRAK) contains all RACF commands. The commands in BBOWBRAK are generated by submitting *hlq*.CNTL(BBOCBRAJ). If you browse *hlq*.DATA(BBOWBRAK), you can examine the generated RACF commands.

These RACF commands are generated:

- The common RACF security setup is required regardless of the security option:
  - Activate RACF SERVER, STARTED, and FACILITY classes.
  - Add user for WebSphere Application Server regions.
  - Add default asynchronous admin task user.
  - Connect servants to the WebSphere configuration group.
  - Define and permit access for the log stream profile.
  - Define and permit access for SERVER CB profiles. Determine whether a servant region can initialize.
  - Define FACILITY BPX.WLMSERVER profiles. Authorize servants to use Workload Manager (WLM) services.
  - Assign user IDs to STARTED tasks.
  - Define permissions to work with certificates.
  - Create a certificate authority (CA) certificate.
  - Create an SSL keyring (WebSphere Application Server CA certificates/commercial CAs) for the location service daemon.
  - Generate a certificate for the location service daemon.
  - Connect a certificate to the keyring for the location service daemon.

- Additional security setup for z/OS product security uses these classes and tasks:
  - Activate RACF CBIND and SURROGAT classes. The SURROGAT class profile relating to the Sync-to-Thread feature is new in WebSphere Application Server V7.0.
  - Create a WebSphere Application Server administrator user ID and an unauthenticated user ID.
  - Set up the RACF APPL class profile.
  - Set up CBIND class profiles. CBIND profiles are created and granted to the configuration group.
  - Set up EJBROLE class profiles:
    - Administrative roles (administrator, monitor, configurator, operator, deployer, and adminSecurityManager) are created, and the administrator user ID is granted the administrator role.
    - Naming roles (CosNamingRead, CosNamingWrite, CosNamingCreate, and CosNamingDelete) are created. Permit the configuration group for WebSphere Application Server (servers and administrators) READ access to all of these profiles.
  - Create an SSL keyring (WebSphere Application Server CA certificates/commercial CAs) for the application server. Digital keyrings are created for the administrator, controller, controller region adjunct, and server user IDs.
  - Generate a certificate. Digital certificates are created for each server controller.
  - Connect the certificate to the keyring.
  - Create a sync-to-thread profile (new in WebSphere Application Server Version 7.0). Refer to Chapter 14, “Security identity propagation” on page 537 for more information.
  - Create a facility class profile for trusted applications (new in WebSphere Application Server Version 7.0).

    A trusted application is an application that requires WebSphere Application Server to build SAF credentials without an authenticator. An example is a task control block (TCB)-level ACEE being created if sync-to-OS-thread is enabled. The facility class profile is BBO.TRUSTEDAPPS.<*domain*>.<*cluster*>. The control region user ID needs to be permitted with access READ.

Tip: Back out RACF definitions. When setting up a multi-node environment, it is convenient to have a simple way to delete the RACF definitions that have been created during the installation process. The following article in the WebSphere Information Center provides a REXX script, which is capable of generating a back out script on the basis of the old BBOxBRAC script:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rsec_racftools.html`

10.5 System Authorization Facility authorization

WebSphere Application Server provides three options for an authorization provider. You can use the built-in (default) authorization provider, an external Java Authorization Contract for Containers (JACC) authorization provider, or SAF as the authorization provider.

When using the LocalOS as a user registry on z/OS, it is possible, though not mandatory, to choose SAF authorization.

Use SAF authorization to have EJBROLE security enforced by the z/OS security product, such as RACF. In that case, SAF EJBROLE profiles are used for user-to-role authorization for both Java EE applications and the role-based authorization requests (naming and administration) that are associated with application server run time. WebSphere Application Server for z/OS V7.0 uses the authorization policy that is stored in the z/OS security product for authorization. Do not be confused by the name EJBROLE. It is used for Java EE roles in both Java EnterpriseBeans and Web applications.

This option is available when your environment contains z/OS nodes only. It is available by selecting **Security** → **Global security** → **External authorization providers** in the administrative console.

Note: If SAF authorization is chosen, WebSphere Application Server ignores the panel for Role to User Mapping in the administrative console, which you can reach by selecting **User and Groups** → **Manage Users** or **User and Groups** → **Manage Groups**.

If an LDAP registry or custom registry is configured and SAF authorization is specified, a mapping to a z/OS principal is required at each login for any protected methods to run. If the authentication mechanism is Lightweight Third Party Authentication (LTPA), we recommend that you update all of the following configuration entries to include a mapping to a valid z/OS principal (such as WEB_INBOUND, RMI_INBOUND, and DEFAULT).

10.5.1 SAF authorization for administrative roles

Table 10-2 shows a description of each administrative role. These roles are only in effect when administrative security is enabled.

Table 10-2 Administrative roles and their descriptions

Role	Description
monitor	This role has the fewest privileges. A monitor is granted for the following tasks: ► View the WebSphere Application Server configuration. ► View the current state of the application server.
configurator	A configurator has monitor privileges, plus the ability to change the WebSphere Application Server configuration. The configurator can perform all the day-to-day configuration tasks. For instance, a configurator can perform the following tasks: ► Create a resource. ► Map an application server. ► Install and uninstall an application. ► Deploy an application. ► Assign users and groups to role mapping for applications.
operator	An operator has monitor privileges, plus the ability to change the runtime state. For example, an operator can complete the following tasks: ► Stop and start the server. ► Monitor the server status in the administrative console. Compared to a configurator, the operator is not capable of changing the configuration.

Role	Description
administrator	An administrator has operator and configurator privileges, plus additional privileges that are granted solely to the administrator role. For example, an administrator can complete the following tasks: ► Modify the server user ID and password. ► Configure authentication and authorization mechanisms. ► Enable or disable administrative security, Java2 security, and application security. ► Create, update, or delete users/groups in the federated repositories configuration. Note: An administrator cannot map users and groups to the administrator roles.
iscadmins	The iscadmins role grants administrator privileges to manage users and groups in federated repositories. For example, a user of the iscadmins role can complete the following tasks: ► Create, update, or delete users in the federated repositories configuration. ► Create, update, or delete groups in the federated repositories configuration.
deployer	This role can perform configuration actions and runtime operations on applications.
adminsecuritymanager	When using **`wsadmin`**, the AdminSecurityManager role is able to map users to administrative roles. Also, when fine-grained admin security is used, users granted this role can manage authorization groups. Refer to the AdminSecurityManager role in 10.8, "Fine-grained security" on page 407 for more details.

Role	Description
auditor	*New in V7.0* An auditor can view and modify the configuration settings for the security auditing subsystem and can complete the following tasks: ► Enable and disable the security auditing subsystem. ► Select the event factory implementation to be used with the event factory plug-in point. ► Select and configure the service provider or emitter. ► Set the audit policy that describes the behavior of the application server in the event of an error with the security auditing subsystem. ► Define which security events are to be audited. The auditor role includes the monitor role, which allows the auditor to view but not change the rest of the security configuration.

Mapping administrative user IDs to roles

When enabling administrative security using the default authorization provider, the primary administrative user ID that you specify is automatically mapped to the administrator role. Additional users and groups are mapped to the administrative roles through the administrative console (WebSphere bindings).

When SAF is your authorization provider, you need to permit users and groups to the EJBROLE administrative profiles. If you have customized your application server environment with the “z/OS security product” option, all administrative role profiles are defined in the SAF EJBROLE class except for the iscadmins role. This role is not needed for SAF authorization, because the purpose of this role is to secure the management of users and groups in federated repositories.

EJBROLE profiles are case-sensitive. If a SAF profile prefix is enabled, the administrative role names need to be prefixed by the domain name `<domain>`. Otherwise, just the role name needs to be specified in the profile:

- `RDEFINE EJBROLE <domain>.administrator UACC(NONE)`
- `RDEFINE EJBROLE <domain>.monitor       UACC(NONE)`
- `RDEFINE EJBROLE <domain>.configurator UACC(NONE)`
- `RDEFINE EJBROLE <domain>.operator     UACC(NONE)`
- `RDEFINE EJBROLE <domain>.deployer     UACC(NONE)`
- `RDEFINE EJBROLE <domain>.adminsecuritymanager UACC(NONE)`
- `RDEFINE EJBROLE <domain>.auditor UACC(NONE)`

An example is `RDEFINE EJBROLE WP.administrator UACC(NONE)`.

Users and groups can be permitted to these roles using the following command:

```
PERMIT <domain>.role_name CLASS(EJBROLE) ID(user_id or group_id)
ACCESS(READ)
```

An example is `PERMIT WP.administrator CLASS(EJBROLE) ID(WPADMIN) ACCESS(READ)`.

Mapping a RACF user or group to a specific administrative role requires READ access to the EJBROLE profile. Instead of granting access to the EJBROLE profile for each user of an application, it is common practice to use RACF groups. Therefore, special RACF groups are created and permitted with READ access to several roles. By connecting users to this group, they automatically inherit the roles of the group.

Important: Be careful with granting ALTER access on a discrete resource profile. A user with ALTER access can alter the access control list (ACL) of that profile. It is sufficient to grant READ access.

The EJBROLE class needs to be refreshed in order to activate changes using the following command:

```
SETROPTS RACLIST(EJBROLE) REFRESH
```

In addition, if SAF authorization is enabled during customization, the APPL profile restricts access to WebSphere Application Server by default. Although a user has permission to an administrative EJBROLE profile, the user cannot log on to the administrative console as long as the user has no READ permission to the APPL profile:

```
RDEFINE APPL <domain> UACC(NONE)
PERMIT <domain> CLASS(APPL) ID(WPADMIN) ACCESS(READ)
SETROPTS RACLIST(APPL) REFRESH
```

All identities using WebSphere Application Server, including administrative identities, user IDs that are granted for a EJBROLE profile, and unauthenticated identities, need access to the APPL profile. If you have not specified a SAF profile prefix, the APPL profile used is `CBS390`. Otherwise, it uses the SAF profile prefix instead. Although the APPL class has been activated, the APPL profile can be turned off by using the administrative console. To turn off the profile, in the following path of the administrative console, select **Global security** → **External authorization providers** → **SAF authorization options**, and clear the check box for **Use the APPL profile to restrict access to the application server**.

10.5.2 SAF authorization for applications

If application security is enabled, the application-specific roles are checked by the application server. The application-specific roles are implemented the same way as the administrative roles using EJBROLE profiles. For every Java EE role that is defined in the deployment descriptor, a corresponding EJBROLE profile containing the identical Java EE role has to be defined.

In Example 10-1, we have modified the deployment descriptor of the demonstration application `PlantsbyWebSphere` and added a new Java EE role, a security constraint, and an authentication method. You can make this modification by using Eclipse-based Rational Application Developer Assemble and Deploy. The Java EE role `PlantsUACC` (marked in bold) needs to be defined in the EJBROLE profile.

Example 10-1 Extract of a deployment descriptor

```
<security-constraint>
      <display-name>PlantsByWebSphere</display-name>
      <web-resource-collection>
         <web-resource-name>PlantsByWebSphere</web-resource-name>
         <url-pattern>/*</url-pattern>
         <http-method>GET</http-method>
         <http-method>POST</http-method>
      </web-resource-collection>
      <auth-constraint>
         <description>
         Application PlantsByWebSphere UACC</description>
         <role-name>PlantsUACC</role-name>
      </auth-constraint>
      <user-data-constraint>
         <transport-guarantee>CONFIDENTIAL</transport-guarantee>
      </user-data-constraint>
</security-constraint>
<login-config>
   <auth-method>BASIC</auth-method>
</login-config>
<security-role>
   <description>Application PlantsByWebSphere UACC</description>
   <role-name>PlantsUACC</role-name>
</security-role>
```

The following EJBROLE profile needs to be defined.

```
RDEFINE EJBROLE <domain>.PlantsUACC UACC(NONE)
```

You can grant the permission to this application-specific role by using the command:

```
PERMIT <domain>.PlantsUACC CLASS(EJBROLE) ID(user_id or group_id)
ACCESS(READ)
```

10.5.3 Displaying EJBROLE profiles

In order to display an EJBROLE profile, including its access control list (ACL), execute the following command:

```
RLIST EJBROLE <domain>.administrative_role ALL
```

In general, we recommend that you use JCL for the execution of RACF commands, because the result can be viewed and documented easily. A sample JCL to list all EJBROLE class profiles is shown in Example 10-2.

Example 10-2 Sample JCL to display all profiles in the EJBROLE class

```
//LSTRACF  EXEC PGM=IKJEFT01
//SYSLBC   DD   DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSTSIN  DD   *
        RLIST EJBROLE * ALL
/*
```

10.5.4 SAF EJB role mapper

WebSphere Application Server for z/OS V7.0 supports the use of a custom SAF EJB role mapper. The *custom SAF EJB role mapper* allows an installation to map Java EE role names to SAF EJBRole profile names. Without the SAF EJB role mapper, you must deploy an application by using a role in the deployment descriptor of a component that is identical to the name of an EJBROLE class profile. The security administrator defines EJBROLE profiles and provides the permission to these profiles to SAF users or groups.

Using SAF EJBROLE class profiles can conflict with the standard Java EE role naming conventions. Java EE role names are Unicode strings of any length. Currently, RACF class profiles are restricted to 240 characters in length and cannot be defined if these profiles contain any white spaces or extended code page characters. If a Java EE role name for an installation conflicts with these

RACF restrictions, an installation can use the SAF EJB role mapper exit to map the desired Java EE role name to an acceptable class profile name.

The custom SAF role mapper is a Java-based exit to replace the EJBROLE class profile construction algorithm. The custom SAF role mapper is called to generate a profile for authorization and delegation requests. The role mapper passes the name of the application, and the name of the role then passes back the appropriate class profile name.

This option is available using the administrative console by selecting **Security** → **Global security** → **External authorization provider**, selecting **SAF authorization options**, and clicking **Configure**.

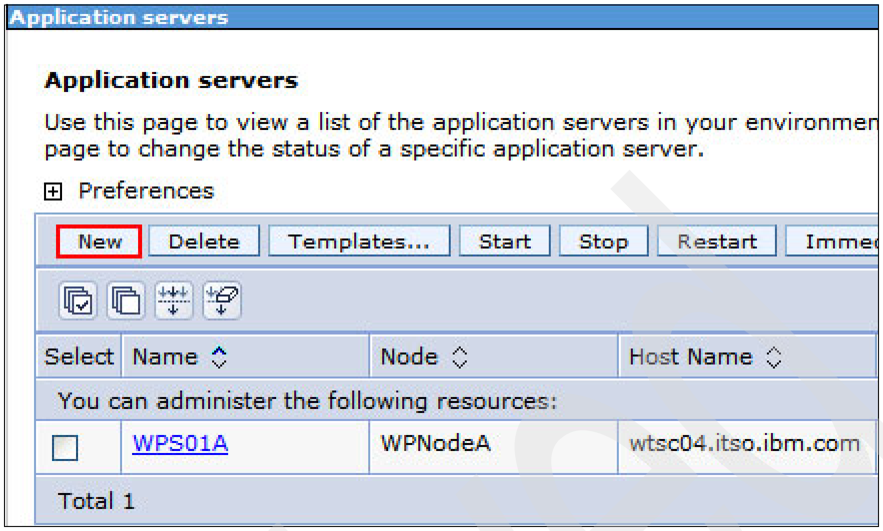
An example for implementing a SAF EJB role mapper can be found in the WebSphere Information Center by following the provided link:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/
com.ibm.websphere.zseries.doc/info/zseries/ae/tsec_safrolemap.html
```

Important: If the SAF delegation is utilized as described in 14.1.3, “SAF delegation” on page 540, the SAF EJB role mapper has an impact on SAF delegation.

10.6 Generic RACF profiles (best practices)

In a Network Deployment environment, additional servers can be created within a node using the wizard in the Application servers section in the administrative console as shown in Figure 10-9 on page 400. But the new server usually cannot be started without defining additional profiles to RACF. In a development environment, it simplifies the administration if additional servers can be created without the need of a RACF administrator.


Application servers

Application servers

Use this page to view a list of the application servers in your environmen
page to change the status of a specific application server.

⊞ Preferences

Select	Name	Node	Host Name
You can administer the following resources:			
☐	WPS01A	WPNodeA	wtsc04.itso.ibm.com
Total 1			

Figure 10-9 Create application server through the administrative console

Generic RACF profiles are the best practice to solve this problem. With the use of wildcards, the RACF profiles can be made generic according to the naming conventions. Every created server that complies with these naming conventions can start successfully. Before starting to work with generic RACF profiles, it is essential to find a naming convention that allows you to extend the number of nodes, as well as the number of servers, without any limitations.

Review the documentation: IBM Washington Systems Center (WSC) experts have summarized their best practices regarding naming conventions in *WebSphere z/OS -- WSC Sample ND Configuration,* WP10653. You can obtain this white paper at the following Web site:

```
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP10065
3
```

In addition, the IBM Techdocs *WebSphere for z/OS Version 7 - Configuration Planning Spreadsheets* have been developed to help you create and document your own naming convention. The spreadsheet asks for several input parameters, such as LPAR name, high-level qualifier (HLQ) for WebSphere, two letter cell prefix, and so forth. The spreadsheet generates a suggestion for your naming convention. Download it from the following Web site:

```
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3341
```

Using generic profiles implies that started task control (STC) user IDs will be reused to a certain extent. Otherwise, you need to create new RACF user IDs and profiles. Use a separate user ID for the application server control and servant regions. But, these application server user IDs can be reused within a node or even across multiple nodes. This reuse depends on your individual security strategy on z/OS as discussed in 10.2, “User ID strategy for a Network Deployment environment” on page 373.

Generic RACF profiles are of interest in a Network Deployment environment, because a base application server is not scalable regarding the number of application servers.

10.6.1 Generic RACF profiles using wildcards

During the customization process of an application server node, the RACF definitions are generated by the Profile Management Tool (PMT) and uploaded to the host. The RACF profile definitions can be found in the BBOWBRAC member of the *hlq*.DATA dataset. In this section, we modify the RACF profiles that are necessary or optional for new application servers.

To make the profiles generic, the use of wildcards is essential. In RACF, either an asterisk (*), percent sign (%), or a combination of both characters can be selected as a wildcard. The asterisk can substitute any number of characters whereas the percent sign can substitute exactly one character. For instance, the `STARTED` profile `WPS%%S.*` accepts the started task name `WPS01AS`, but not `WPS001AS`. If the STC name `WPS001AS` must be accepted by RACF, the profile must be defined as `WPS*S.*` or as `WPS%%%S` instead.

If you have applied a naming convention according to the rules of the white paper *WebSphere z/OS -- WSC Sample ND Configuration,* WP10653, a good starting point is to examine the BBOWBRAC member of the occurrence of “`01`”. This BBOWBRAC member of the occurrence of “`01`” needs to be part of the short name of the first application server, for instance, `WPS01A`. Within the profile definitions, `01` can be substituted by the wildcard %%. Additional servers on the same node can be created by increasing this number, for instance, `WPS02A`. Be careful with the **`change all`** command, because this string is normally part of the certificate expiry date. This action covers most changes in BBOWBRAC, which we describe in detail in this section. The RACF definitions of the deployment manager in BBODBRAC do not need to be changed, because the application server-related definitions are all in BBOWBRAC.

In this example, we have applied a naming convention using the two character cell abbreviation `WP`, and the control region (CR) and servant region (SR) user IDs of the application server will be reused across multiple nodes.

The following profile assigns, for instance, the user ID WPACR to the proclib member WPACRA. According to the WSC naming convention, the last character, which is `A`, is the node identifier. If the control region (CR) user ID of the application server is reused across multiple nodes, WPACRA can be substituted by WPACR%.

```
RDEFINE STARTED WPACR%.* STDATA(USER(WPACR)
GROUP(WPCFG) TRACE(YES))
```

Because the *control region adjunct (CRA)* and the servant region (SR) will be started by the CR, the user IDs are dependent on the STC name. For instance, the following profiles assign the user ID WPACR to STC name WPS01AA (control region adjunct) and WPASR to STC name WPS01AS (servant region).

```
RDEFINE STARTED WPS%%%A.* STDATA(USER(WPACR)
GROUP(WPCFG) TRACE(YES))

RDEFINE STARTED WPS%%%S.* STDATA(USER(WPASR)
GROUP(WPCFG) TRACE(YES))
```

The STC name is structured according to the WSC naming convention, as shown in Table 10-3.

Table 10-3 STC name structure

Cell prefix	**Server prefix**	**Server number**	**Node identifier**	**Region type (SR = S and CRA = A)**
WP	S	01	A	S

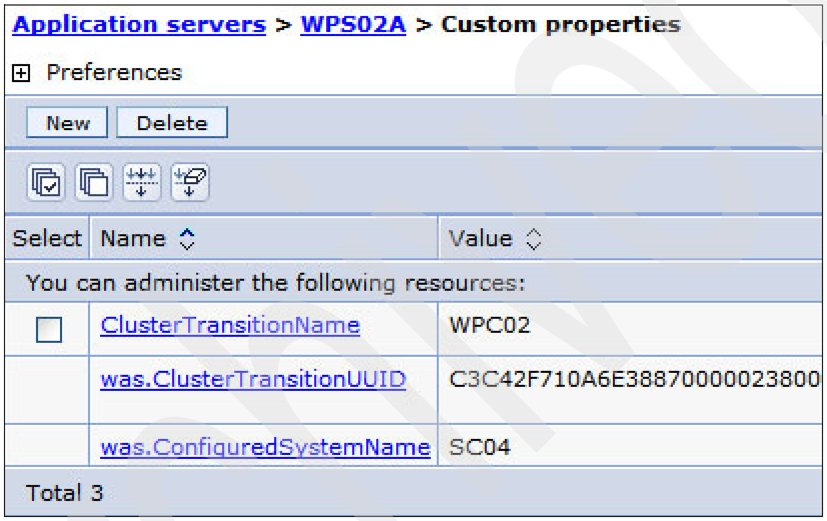
The first two percent signs `WPS%%` of the profiles ensure that the server number can be increased. The third percent sign `WPS%%%` represents the node identifier. Only make the node identifier generic if the application server CRA and CR user IDs are reused across multiple nodes.

The SERVER class is used to control whether a servant can call authorized programs in the controller and whether the corresponding profile is structured in the following qualifiers: `CB.<server>.<cluster>.<cell>`.

```
RDEFINE SERVER CB.*.WPC%%ADJUNCT.* UACC(NONE)
RDEFINE SERVER CB.*.WPC%%.* UACC(NONE)

PERMIT CB.*.WPC%%.* CLASS(SERVER) ID(WPACR) ACC(READ)
PERMIT CB.*.WPC%%.* CLASS(SERVER) ID(WPSRG) ACC(READ)
PERMIT CB.*.WPC%%ADJUNCT.* CLASS(SERVER) ID(WPACR) ACC(READ)
```

The third qualifier `<cluster>` is the cluster name if a cluster has been defined. The cluster transition name is used instead for an application server that is not a member of a cluster. The cluster transition name is a variable called `ClusterTransitionName` in the custom properties of a selected application server in **Application servers** → *app_server_name* → **Administration** → **Custom properties** as shown in Figure 10-10. According to the naming convention of the WSC, the cluster name usually corresponds to the server names. For additional clusters, the number will be increased. For instance, application servers WPS01A on node A and WPS01B on node B are cluster members of WPC01. By substituting the cluster number with the wildcard `%%`, new servers and clusters can be added without additional RACF profiles.


Application servers > WPS02A > Custom properties

Select	Name	Value
You can administer the following resources:		
☐	ClusterTransitionName	WPC02
	was.ClusterTransitionUUID	C3C42F710A6E38870000023800
	was.ConfiguredSystemName	SC04
Total 3		

Figure 10-10 Location of the cluster transition name

The `BBO.SYNC` profile enables the sync-to-thread capabilities in order to pass user IDs from WebSphere Application Server to DB2. Refer to Chapter 14, “Security identity propagation” on page 537 for more information about this topic. If the cluster number, which is part of the cluster name or cluster transition name, is substituted by the wildcard `%%`, Sync-to-Thread can be managed with one profile for the complete cell, instead of one profile for each server or cluster:

```
RDEFINE FACILITY BBO.SYNC.WPCELL.WPC%% UACC(NONE)
```

The `BBO.TRUSTEDAPPS` profile controls the ability of an application server to change the thread identity of execution from its own STC user ID to the Java EE identity of the user. We recommend that you enable Trusted Application for SAF authorization, which is critical to the z/OS integrity statement. Therefore, the configuration group has `READ` permissions to this profile. To make the `BBO.TRUSTEDASPPS` profile generic, the cluster number, which is part of the cluster name or cluster transition name, can be substituted with the wildcard %%. Consequently, new servers and clusters can be added without additional RACF profiles:

```
RDEFINE FACILITY BBO.TRUSTEDAPPS.WPCELL.WPC%% UACC(NONE)

PERMIT BBO.TRUSTEDAPPS.WPCELL.WPC%% CLASS(FACILITY) ID(WPCFG)
ACCESS(READ)
```

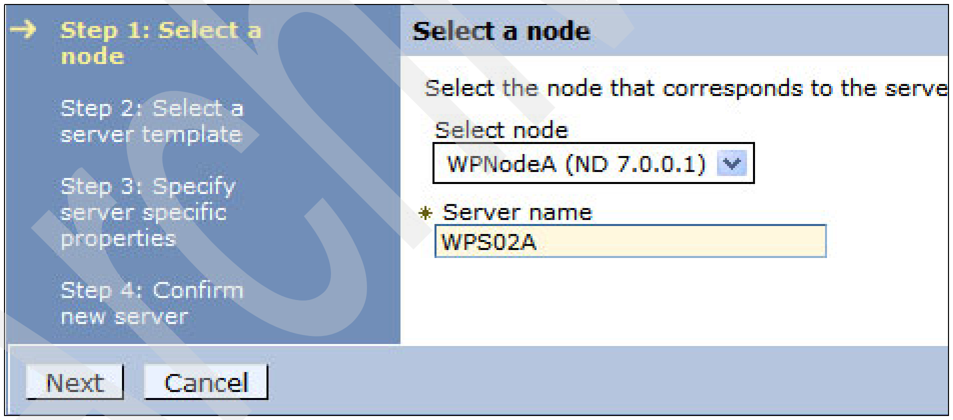
Finally, you can execute the modified definitions by submitting the corresponding BBOCBRAK member in the *hlq*.CNTL dataset.

By implementing these generic RACF profiles in a Network Deployment environment, new servers and nodes can be added without needing additional RACF profiles.

10.6.2 Creating a new server with the administrative console

You can create a new application server by using the appropriate wizard in the administrative console:

1. Select **Application servers** and click **New** to start the wizard (Figure 10-11).


Step 1: Select a node
Step 2: Select a server template
Step 3: Specify server specific properties
Step 4: Confirm new server
Select a node
Next
Cancel

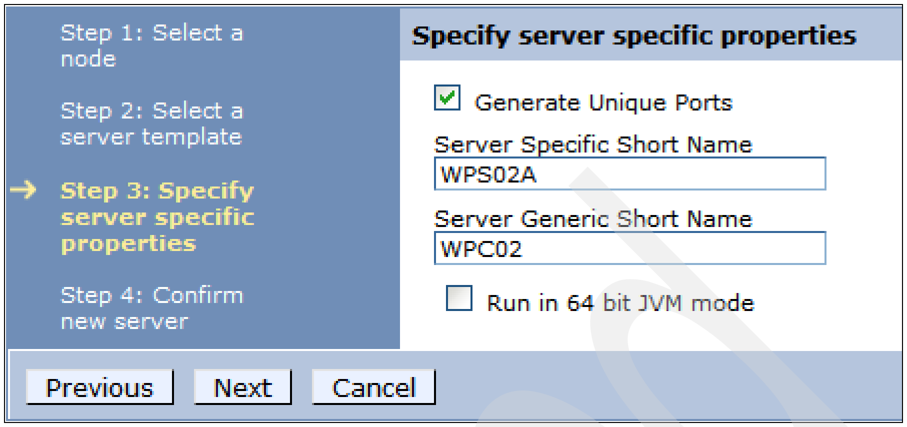
Select the node that corresponds to the serve

Select node: WPNodeA (ND 7.0.0.1)

* Server name: WPS02A

Figure 10-11 Assign the node and enter the server name

2. Assign the new application server to a node, and specify the server long name according to your naming conventions.
3. Select the default z/OS template.
4. In Figure 10-12 on page 405, select **Generate Unique Ports**, and specify the server short name and server generic short name (which is the same name as the cluster transition name) according to your naming convention.


Step 1: Select a node
Step 2: Select a server template
Step 3: Specify server specific properties
Step 4: Confirm new server
Previous
Next
Cancel

Specify server specific properties

[✓] Generate Unique Ports

Server Specific Short Name: WPS02A

Server Generic Short Name: WPC02

[ ] Run in 64 bit JVM mode

Figure 10-12 Assign the server short name and cluster transition name

5. Click **Finish**, save, and synchronize the changes with the nodes.

If the generic RACF profiles have been implemented correctly, you can start the new application server from the administrative console.

10.7 Case-sensitive passwords for RACF

The password is important to protect users, applications, and systems. The harder it is to guess a password, the better the protection level. Mixed case passwords provide a higher level of security and are harder to guess.

WebSphere for z/OS V7.0 supports multiple user registries. Both file-based and LDAP repositories support case-sensitive passwords. RACF mixed-case password support was introduced in z/OS V1.7. WebSphere Application Server for z/OS V7.0 supports RACF mixed-case passwords. The WebSphere support applies both to a local OS user registry and LDAP DB2 Technical Database Management (TDBM) native authentication or SDBM. Passwords are validated in RACF in both LDAP types.

RACF mixed-case password support is disabled by default. To enable mixed-case password support, you must comply with the following prerequisites:

- WebSphere Application Server for z/OS V6.1 or later is required.
- z/OS Version 1.7 or later is required.
- In WebSphere, either local operating system registry, LDAP TDBM NA, or LDAP SDBM is the configured registry.

The SETROPTS option to enable or disable mixed-case passwords is not available through the RACF panels. You need to issue one of the following commands.

To turn on the MIXEDCASE option (Figure 10-13), enter this command:

```
SETROPTS PASSWORD(MIXEDCASE)
```

To turn off the MIXEDCASE option, enter this command:

```
SETROPTS PASSWORD(NOMIXEDCASE)
```

To display the current system-wide RACF options for your environment, issue this command:

```
SETROPTS LIST
```

```
PASSWORD PROCESSING OPTIONS:
  PASSWORD CHANGE INTERVAL IS  90 DAYS.
  PASSWORD MINIMUM CHANGE INTERVAL IS   0 DAYS.
  MIXED CASE PASSWORD SUPPORT IS IN EFFECT
  NO PASSWORD HISTORY BEING MAINTAINED.
  USERIDS NOT BEING AUTOMATICALLY REVOKED.
  PASSWORD EXPIRATION WARNING LEVEL IS  10 DAYS.
  NO INSTALLATION PASSWORD SYNTAX RULES ARE PRESENT.
```

Figure 10-13 Password excerpt of the SETROPTS LIST command output

Important: Carefully plan the use of mixed-case passwords on z/OS systems. Instructing your users to use mixed-case passwords is of utmost importance to avoid confusion after a user has entered a new password. Falling back to NOMIXEDCASE has an even bigger impact, because all mixed-case passwords must be reset.

Pass phrase passwords for RACF

Another possibility to increase the number of password combinations is to use pass phrases, which were introduced with z/OS V1.9. A normal RACF password is usually restricted to eight characters. In contrast, a pass phrase can be from nine to 100 characters long, which gives you an exponentially larger number of combinations for securing any given user ID to an application. A user ID can have both a password and a password phrase associated with it. The user ID uses the password for existing applications that accept an eight-character password and the password phrase for those applications that are sensitive to the longer character string.

In order to implement pass phrases for WebSphere Application Server for z/OS, you must have the following requirements:

- z/OS V1.9 or higher is required.
- You must select localOS as the user repository and SAF as the authorization provider.
- WebSphere Application Server Fix Pack 6.1.0.15 or later must be installed.
- If you want to specify a password phrase that is between nine and 13 characters, inclusive, you must also install the ICHPWX11 RACF exit routine.

After a restart of the cell, WebSphere Application Server accepts pass phrases for authentication.

It is possible to combine pass phrases with the RACF MIXEDCASE option.

10.8 Fine-grained security

In previous WebSphere versions, administrative roles allowed users to administer all WebSphere artifacts or resource instances (cell, nodes, servers, clusters, and applications) in the realm of the cell. WebSphere Application Server V7.0 fine-grained administrative security gives you the capability to define scopes of authorization at, for instance, the cell, node, or server level.

The *authorization group* outlines the scope of authorization. Users can then be granted an administrative role in an authorization group. In this way, users are only allowed to administer a delimited scope of resources.

The following *resource instances* can be added to an authorization group:

- Cell
- Node
- Server cluster
- Server
- Application
- NodeGroup

A resource instance can only belong to one authorization group.

The administrative roles are granted per authorization group and therefore per resource instance rather than to the entire cell. However, there is a cell-wide authorization group for compatibility with previous versions. Users assigned to administrative roles in the cell-wide authorization group can still access all of the resources within the cell.

Fine-grained administrative security can also be used in single-server environments. Various applications in the single server can be grouped and placed in separate authorization groups. The stand-alone server itself cannot be part of any authorization group.

The AdminSecurityManager role is only available through **`wsadmin`**. Users granted this role can map users to administrative roles. When fine-grained administrative security is used, users granted this role can manage authorization groups.

There is no on-off switch to activate fine-grained administrative security. After you set up an authorization group, it is active. In WebSphere Application Server for z/OS, there are two ways to implement fine-grained administrative security control, which is the same in standard administrative security. You can implement fine-grained administrative security control through WebSphere application security, which is used when selecting the default authorization provider. You can also implement fine-grained administrative security control through RACF, which is used when SAF is the authorization provider.

Fine-grained access is granted by performing the following steps:

1. Connect to the application server with **`wsadmin`**.
2. Create an authorization group.
3. Add resource instances to the authorization group.
4. Map users and groups to administrative roles at the authorization group level. Users and groups need to exist in the configured user registry.
5. Save the changes.
6. Restart the cell to activate the changes.

Steps 1 to 3, 5, and 6 are common for all authorization providers.

Step 4 varies according to the chosen external authorization provider.

The following scenario illustrates how to map users to roles for both SAF authorization with the z/OS security product and for default authorization with WebSphere Application Server security.

In the scenario, a user is granted deployer access to a specific resource (an application). This user cannot administer any other resources outside of the user's authorization group. Users assigned to administrative roles in the cell-wide authorization group can still access all of the resources within the cell.

We used these values to build this scenario:

- Users, groups, and resource instance:
  - Administrator role user: `WPADMIN` is assigned to the cell level administrator role to create an authorization group, add resources to the authorization group, and so forth.
  - Authorization group: `PLANTS`.
  - Resource instance: An application named `PlantsByWebSphere` is added to `PLANTS`.
  - Security role user: `APPUSER1` is mapped to the `deployer` role in the `PLANTS` authorization group.
- Server environment:
  - Cell name: `WPCell`
  - Node name: `WPNodeA`
  - Server name: `WPS01A`
  - server_profile_root: `/wasconfig/wpcell/wpdmnode/DeploymentManager/profiles/default/bin`

Follow these steps:

1. Connect to the application server with **wsadmin**.
2. Create an authorization group.
3. Add resource instances to the authorization group.

Example 10-3 shows the commands and the output of steps 1 to 3.

Example 10-3 Execution output of creating new security group

```
SC04:/wasconfig/wpcell/wpdmnode/DeploymentManager/profiles/default/bin>
wsadmin.sh -lang jython -user WPADMIN -password wsadmin
WASX7209I: Connected to process "dmgr" on node WPDmNode using SOAP
connector;  The type of process is: DeploymentManager
WASX7031I: For help, enter: "print Help.help()"
wsadmin>AdminTask.createAuthorizationGroup('[-authorizationGroupName
PLANTS]')
'cells/WPCell/authorizationgroups/PLANTS|authorizationgroup.xml#Authori
zationGroup_1236045756589'
wsadmin>AdminTask.addResourceToAuthorizationGroup('[-authorizationGroup
Name PLANTS -resourceName Application=PlantsByWebSphere]')
''
```

4. Map users and groups to roles at the authorization group scope.

   If you use the default authorization provider as the authorization provider, issue this command (Example 10-4 shows the output of this command):

   ```
   AdminTask.mapUsersToAdminRole ('[-authorizationGroupName PLANTS
   -roleName deployer -userids APPUSER1]')
   ```

Example 10-4 Execution output of role mapping

```
wsadmin>AdminTask.mapUsersToAdminRole ('[-authorizationGroupName PLANTS
-roleName deployer -userids APPUSER1]')
'true'
```

5. If you use the SAF authorization provider as the authorization provider, define additional profiles to the EJBROLE class:

   a. Define <*authorization group*>.<*rolename*> profiles in EJBROLE (again, prefix those profiles with your domain identifier if you have one configured):

      ```
      RDEFINE EJBROLE <domain>.PLANTS.administrator UACC(NONE)

      RDEFINE EJBROLE <domain>.PLANTS.configurator UACC(NONE)

      RDEFINE EJBROLE <domain>.PLANTS.operator UACC(NONE)

      RDEFINE EJBROLE <domain>.PLANTS.monitor UACC(NONE)

      RDEFINE EJBROLE <domain>.PLANTS.deployer UACC(NONE)

      RDEFINE EJBROLE <domain>.PLANTS.adminsecuritymanager UACC(NONE)

      RDEFINE EJBROLE <domain>.PLANTS.auditor UACC(NONE)
      ```

      Note: You must create EJBROLE profiles for all of the roles in each authorization group regardless of whether any user is mapped to its role. Or, at least define an appropriate catching profile at the authorization group level.

   b. Permit the user to access the EJBROLE class profile:

      ```
      PERMIT <domain>.PLANTS.deployer CLASS(EJBROLE) ID(APPUSER1)
      ACCESS(READ)
      ```

   c. In order to effectively use the administrative console, this user must have a cell-wide monitor role.

      ```
      PERMIT <domain>.monitor CLASS(EJBROLE) ID(APPUSER1) ACCESS(READ)
      ```

   d. Refresh the RACLISTed class to activate the changes:

      ```
      SETROPTS RACLIST(EJBROLE) REFRESH
      ```

6. Save the changes using **wsadmin** (Example 10-5 shows the output of this command):

   ```
   AdminConfig.save()
   ```

Example 10-5 Execution output of save command

```
wsadmin>AdminConfig.save()
''
```

7. Restart the application server to activate the changes.

APPUSER1 is assigned the deployer role for the PlantsByWebSphere demonstration application, but APPUSER1 does not have administrative authority for other applications. In Example 10-6, you can see that WPADMIN can list and control all deployed applications.

Example 10-6 WPADMIN execution output

```
SC04:/wasconfig/wpcell/wpdmnode/DeploymentManager/profiles/default/bin>
wsadmin.sh -lang jython -user WPADMIN -password wsadmin
WASX7209I: Connected to process "dmgr" on node WPDmNode using SOAP
connector;  The type of process is: DeploymentManager
WASX7031I: For help, enter: "print Help.help()"
wsadmin>print AdminApp.list()
DefaultApplication
Dynamic Cache Monitor
FRCA_App
PlantsByWebSphere
SamplesGallery
Trade6
ivtApp
query
wsadmin>appManager =
AdminControl.queryNames('cell=WPCell,node=WPNodeA,type=ApplicationManag
er,process=WPS01A,*')
wsadmin>AdminControl.invoke(appManager,'stopApplication','PlantsByWebSp
here')
''
wsadmin>AdminControl.invoke(appManager,'stopApplication','DefaultApplic
ation')
''
wsadmin>
```

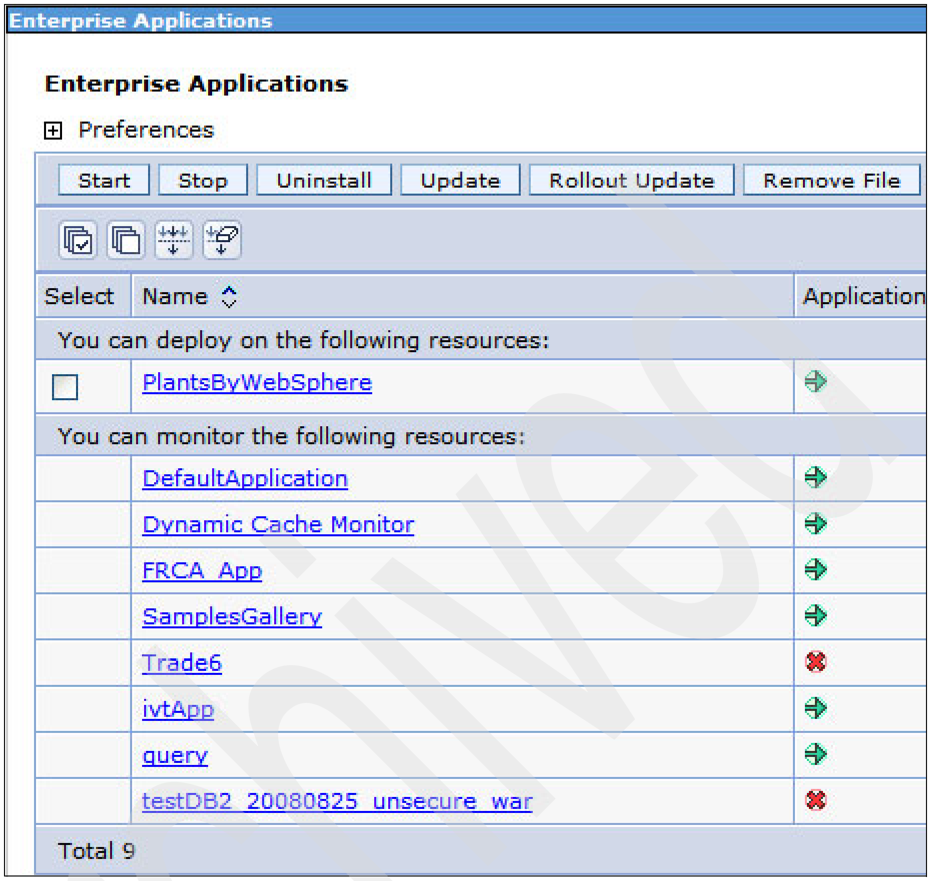
In Example 10-7 on page 412, you can see that APPUSER1 can list and control only the authorized resource.

Example 10-7 APPUSER1 execution output

```
SC04:/wasconfig/wpcell/wpdmnode/DeploymentManager/profiles/default/bin>
wsadmin.sh -lang jython -user APPUSER1 -password pswd
WASX7209I: Connected to process "dmgr" on node WPDmNode using SOAP
connector;  The type of process is: DeploymentManager
WASX7031I: For help, enter: "print Help.help()"
wsadmin>print AdminApp.list()
PlantsByWebSphere
wsadmin>appManager =
AdminControl.queryNames('cell=WPCell,node=WPNodeA,type=ApplicationManag
er,process=WPS01A,*')
wsadmin>AdminControl.invoke(appManager,'stopApplication','PlantsByWebSp
here')
''
wsadmin>AdminControl.invoke(appManager,'stopApplication','DefaultApplic
ation')
WASX7015E: Exception running command:
"AdminControl.invoke(appManager,'stopApplication','DefaultApplication')
"; exception information:
 javax.management.JMRuntimeException: ADMN0022E: Access is denied for
the stopApplication operation on ApplicationManager MBean because of
insufficient or empty credentials.

wsadmin>
```

Figure 10-14 on page 413 shows how fine-grained security is handled by the administrative console. A cell-wide monitor role is required to use the administrative console.


Enterprise Applications

Enterprise Applications

⊞ Preferences

Select	Name	Application
You can deploy on the following resources:		
☐	PlantsByWebSphere	➜
You can monitor the following resources:		
	DefaultApplication	➜
	Dynamic Cache Monitor	➜
	FRCA_App	➜
	SamplesGallery	➜
	Trade6	✖
	ivtApp	➜
	query	➜
	testDB2_20080825_unsecure_war	✖
Total 9		

Figure 10-14 Fine-grained security in the administrative console

For more information about the AdminTask object that is related to fine-grained security, refer to the WebSphere Information Center. Obtain more information about implementing fine-grained security in TechDoc TD103324:

`http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD103324`

10.9 Naming service security

The naming service is used by clients of WebSphere Application Server applications to obtain references to objects related to these applications. These objects are bound into a mostly hierarchical structure, referred to as a *name space*. The name space structure consists of a set of name bindings, each consisting of a name relative to a specific context and the object bound with that name.

The concept of role-based authorization has been extended to protect the WebSphere Common Object Request Broker Architecture (CORBA) naming service (CosNaming).

10.9.1 CosNaming roles

CosNaming security offers increased granularity of security control. CosNaming functions are available on CosNaming servers. These functions affect the content of the WebSphere Application Server name space.

You can access and manipulate the name space through a *name server*. Users of a name server are referred to as *naming clients*. Naming clients typically use two interfaces:

- Java Naming and Directory Interface (JNDI)
- Common Object Request Broker Architecture (CORBA) naming interface

You can control access to the name space by using CosNaming roles. All CosNaming roles are shown in Table 10-4 on page 415.

Table 10-4 CosNaming security roles description

Role	Description
CosNamingRead	Query the WebSphere Application Server name space using, for example, the JNDI lookup method. The EVERYONE special-subject is the default policy for this role.
CosNamingWrite	Perform write operations, such as JNDI bind, rebind, or unbind, and CosNamingRead operations. The ALL_AUTHENTICATED special-subject is the default policy for this role.
CosNamingCreate	Create new objects in the name space through such operations as JNDI createSubcontext and CosNamingWrite operations. The ALL_AUTHENTICATED special-subject is the default policy for this role.
CosNamingDelete	Destroy objects in the name space, for example, using the JNDI destroySubcontext method and CosNamingCreate operations. The ALL_AUTHENTICATED special-subject is the default policy for this role.

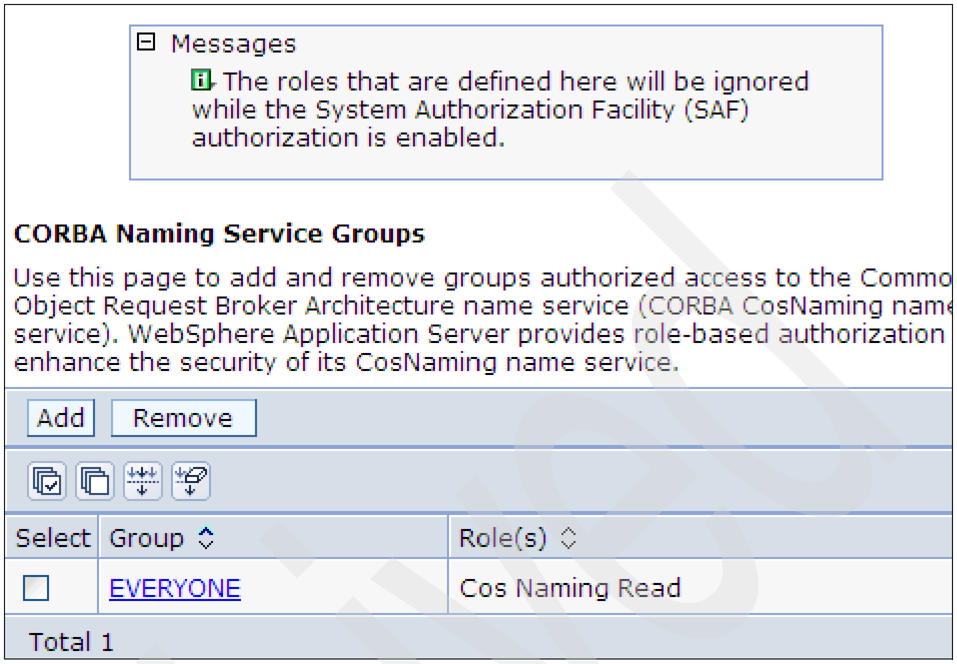
The CosNaming authorization policy is valid only when administrative security is enabled. When administrative security is enabled, attempts to do CosNaming operations without the proper role assignment result in an org.omg.CORBA.NO_PERMISSION exception from the CosNaming server.

10.9.2 Mapping users or groups to CosNaming roles

There are two ways to map users or groups to CosNaming roles: through the administrative console or through SAF EJBROLE profiles, depending on the authorization provider setting.

In the administrative console, CosNaming role mapping can be defined at the following panels: **Environment** → **Naming** → **CORBA Naming Service Users** or **CORBA Naming Service Groups**.

Figure 10-15 on page 416 shows a default role definition granting read access to EVERYONE.


Messages

The roles that are defined here will be ignored while the System Authorization Facility (SAF) authorization is enabled.

CORBA Naming Service Groups

Use this page to add and remove groups authorized access to the Commo Object Request Broker Architecture name service (CORBA CosNaming name service). WebSphere Application Server provides role-based authorization enhance the security of its CosNaming name service.

Select	Group	Role(s)
☐	EVERYONE	Cos Naming Read

Total 1

Figure 10-15 Default setting of the CosNamingRead role

Users, groups, or the special-subjects ALL_AUTHENTICATED and EVERYONE can be added or removed at any time. However, a server restart is required for the changes to take effect.

Role mappings added through the administrative console are ignored if SAF authorization is enabled. If you use SAF as the authorization provider, access to the naming role is controlled by RACF. These are the commands to define the naming profiles in the EJBROLE class. Prefix the profiles with your SAF profile prefix identifier if you have one configured.

```
RDEFINE EJBROLE <domain.>CosNamingRead   UACC(READ)
PERMIT <domain.>CosNamingRead  CLASS(EJBROLE)  ID(WPGUEST) ACCESS(READ)
RDEFINE EJBROLE <domain.>CosNamingWrite  UACC(NONE)
RDEFINE EJBROLE <domain.>CosNamingCreate UACC(NONE)
RDEFINE EJBROLE <domain.>CosNamingDelete UACC(NONE)
PERMIT <domain.>CosNamingWrite  CLASS(EJBROLE)  ID(WPCFG) ACCESS(READ)
PERMIT <domain.>CosNamingCreate  CLASS(EJBROLE) ID(WPCFG) ACCESS(READ)
PERMIT <domain.>CosNamingDelete  CLASS(EJBROLE) ID(WPCFG) ACCESS(READ)
SETROPTS RACLIST(EJBROLE) REFRESH
```

In the previous example, WPGUEST is the unauthenticated user ID and WPCFG is the WebSphere configuration group name defined in the server configuration process. CosNamingRead is defined with UACC(READ), which is the closest thing in RACF to the EVERYONE special-subject. Note that we permit the unauthenticated user ID specifically to the CosNamingRead role because it has been defined with the RESTRICTED attribute. The server special-subject WPCFG should be permitted to all of the four CosNaming roles, because it provides processes that run under the server identity access to all the CosNaming operations.

We recommend mapping groups or one of the special-subjects to naming roles. That is more flexible than mapping specific users and is easier to administer. When SAF authorization is the chosen authorization provider, you benefit from the fact that servers do not need to be restarted in order to activate changes in bindings to the naming roles.

11

Administrative security

We strongly recommend that you enable administrative security to restrict access to the administrative console. It also secures the TCP/IP connectivity between WebSphere components with Secure Sockets Layer (SSL). The following sections explain the necessary steps to enable administrative security, assuming that administrative security has not been enabled during the customization process.

This chapter contains the following sections:

Table 11-1 on page 420 shows several products that you can use to implement administrative security.

Table 11-1 Administrative security

	WebSphere security	**z/OS security product**
Realm	Federated repository	Local operating system (OS)
Authorization	WebSphere (default)	System Authorization Facility (SAF) authorization
SSL configuration	Hierarchical file system (HFS) keystore/ truststores	SAF keyring
BBOBRAK (additional RACF commands)	None	Certificates Keyrings CBIND EJBROLE Sync-to-thread EnableTrustedApps

11.1 Selecting an authorization provider

You need to select an authorization provider when enabling administrative security. The authorization information determines whether a user or group has the necessary privileges to access a resource. WebSphere Application Server for z/OS supports the following authorization providers:

- *SAF authorization* uses the SAF EJBROLE class to control client access to Java Platform, Enterprise Edition (Java EE) roles in Enterprise JavaBeans (EJBs) and Web applications.
- *Built-in authorization* (formerly known as default authorization) does not require special setup. Usually, the Java EE roles, user-to-role, and group-to-role mappings are defined in the deployment descriptor of an application and are mapped during run time by the built-in authorization provider. The Java EE roles for the administrative console are an exception. The administrative roles are defined in the administrative console (Users and Groups options).
- *External authorization using a Java Authorization Contract for Containers (JACC) provider.* The JACC-based authorization provider enables third-party security providers, such as Tivoli Access Manager, to handle the J2EE authorization.

Table 11-2 on page 422 summarizes these options.

Table 11-2 Authorization provider characteristics

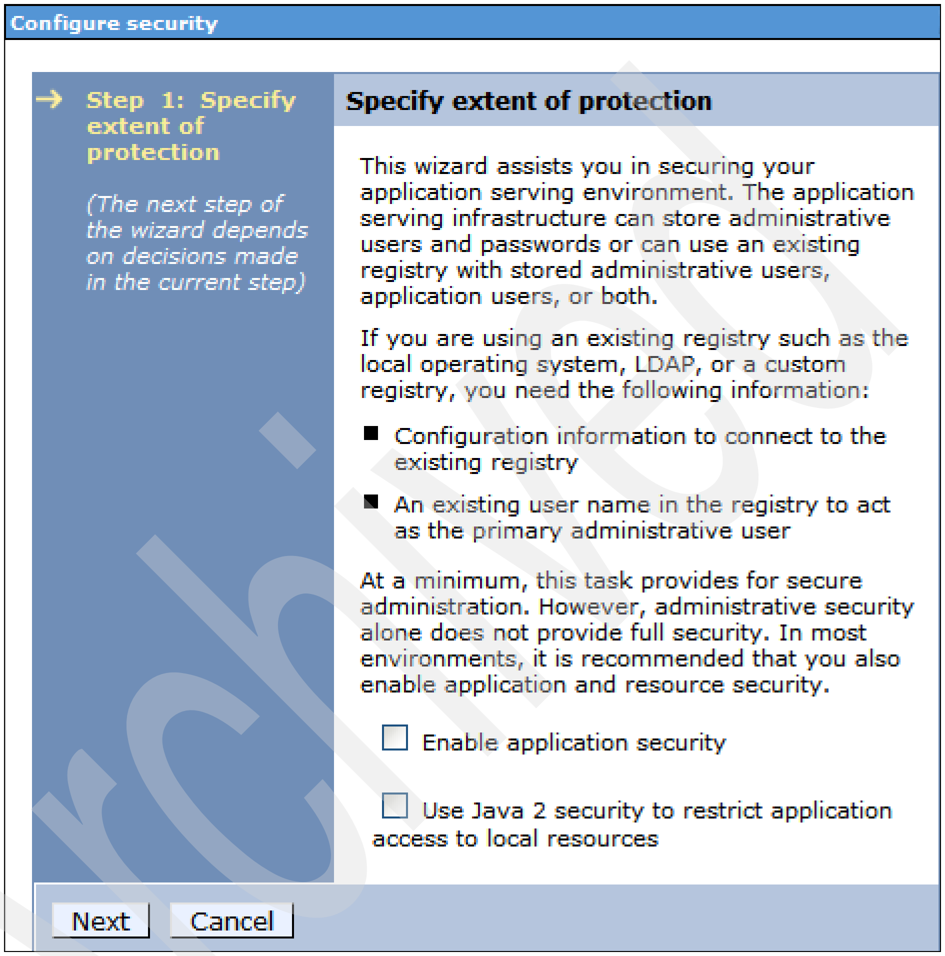
Authorization providers	Characteristics
Built-in authorization (default authorization)	► Access to servlets or EJB methods is based upon the role (job title, function, and so on) of the user or caller. ► Roles are associated with servlets or EJBs at assembly time. ► Roles are stored in the application's .ear file: `application.xml`. ► The users and groups and their associated roles are also stored in the application's .ear file: `ibm-application-bnd.xmi`. ► Roles are managed by the application developer and the application deployer. ► SAF provides user and group information.
SAF authorization	► Access to servlets or EJB methods is based upon the role (job title, function, and so on) of the user or caller. ► Roles are associated with servlets or EJBs at assembly time. ► Roles are represented in the application's .ear file: `application.xml`. ► The users and groups and their associated roles are determined in SAF by profiles in the EJBROLE class. ► If a user is in the access control list (ACL) of an EJBROLE profile, the user has that role. ► If a group is in the ACL of an EJBROLE profile, users in that group have that role. ► Roles are managed through SAF.
External authorization using a JACC provider	► Access to servlets or EJB methods is based upon the role (job title, function, and so on) of the user or caller. ► Roles are associated with servlets or EJBs at assembly time. ► Roles are represented in the application's .ear file: `application.xml`. ► The users and groups and their associated roles are determined in the JACC provider. ► Roles are managed through the JACC provider.

11.2 Enabling security with a SAF user repository

Follow these steps to enable administrative security with SAF as the user repository:

1. Log on to the administrative console.
2. Click **Security** → **Global Security**.
3. Click **Security Configuration Wizard**. Figure 11-1 on page 423 appears:
   a. Specify the extent of the protection. At this point, application security and Java 2 security can be enabled in addition to administrative security. If application security is enabled, the roles defined in the deployment descriptor will be activated, and the authorization provider will verify the

roles that are assigned to a client user. Using SAF authorization provider, the role profiles, as well the user-to-profile mappings, must be defined in SAF.


Configure security

→ Step 1: Specify extent of protection

(The next step of the wizard depends on decisions made in the current step)

Specify extent of protection

This wizard assists you in securing your application serving environment. The application serving infrastructure can store administrative users and passwords or can use an existing registry with stored administrative users, application users, or both.

If you are using an existing registry such as the local operating system, LDAP, or a custom registry, you need the following information:

- Configuration information to connect to the existing registry
- An existing user name in the registry to act as the primary administrative user

At a minimum, this task provides for secure administration. However, administrative security alone does not provide full security. In most environments, it is recommended that you also enable application and resource security.

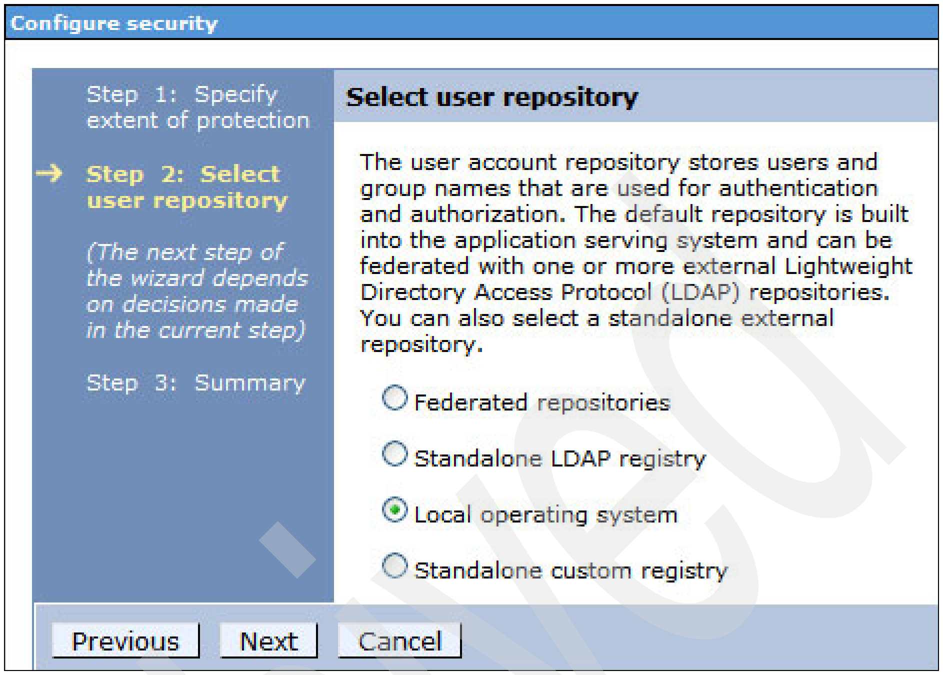
☐ Enable application security

☐ Use Java 2 security to restrict application access to local resources

Figure 11-1 Step 1: Specify extent of protection

Click **Next**.

b. Select **Local operating system** as the repository for users and groups (refer to Figure 11-2 on page 424).


Configure security
Step 1: Specify extent of protection
Step 2: Select user repository
(The next step of the wizard depends on decisions made in the current step)
Step 3: Summary
Previous
Next
Cancel

Select user repository

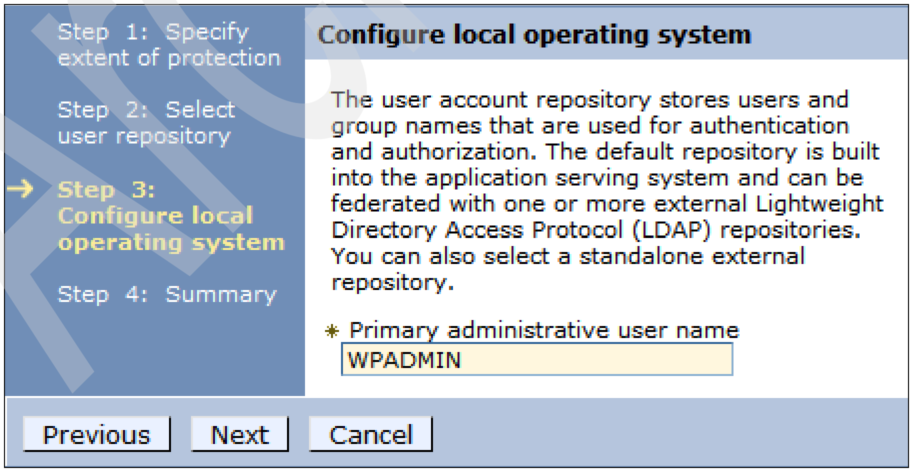
The user account repository stores users and group names that are used for authentication and authorization. The default repository is built into the application serving system and can be federated with one or more external Lightweight Directory Access Protocol (LDAP) repositories. You can also select a standalone external repository.

- ○ Federated repositories
- ○ Standalone LDAP registry
- ● Local operating system
- ○ Standalone custom registry

Figure 11-2 Step 2: Select user repository

Click **Next**.

c. Enter the primary administrative user ID (refer to Figure 11-3). This user has to be defined to the SAF provider.


Step 1: Specify extent of protection
Step 2: Select user repository
Step 3: Configure local operating system
Step 4: Summary
Previous
Next
Cancel

Configure local operating system

The user account repository stores users and group names that are used for authentication and authorization. The default repository is built into the application serving system and can be federated with one or more external Lightweight Directory Access Protocol (LDAP) repositories. You can also select a standalone external repository.

* Primary administrative user name: WPADMIN

Figure 11-3 Step 3: Configure local operating system

Click **Next**.

d. Verify your settings in the summary, and click **Finish**.

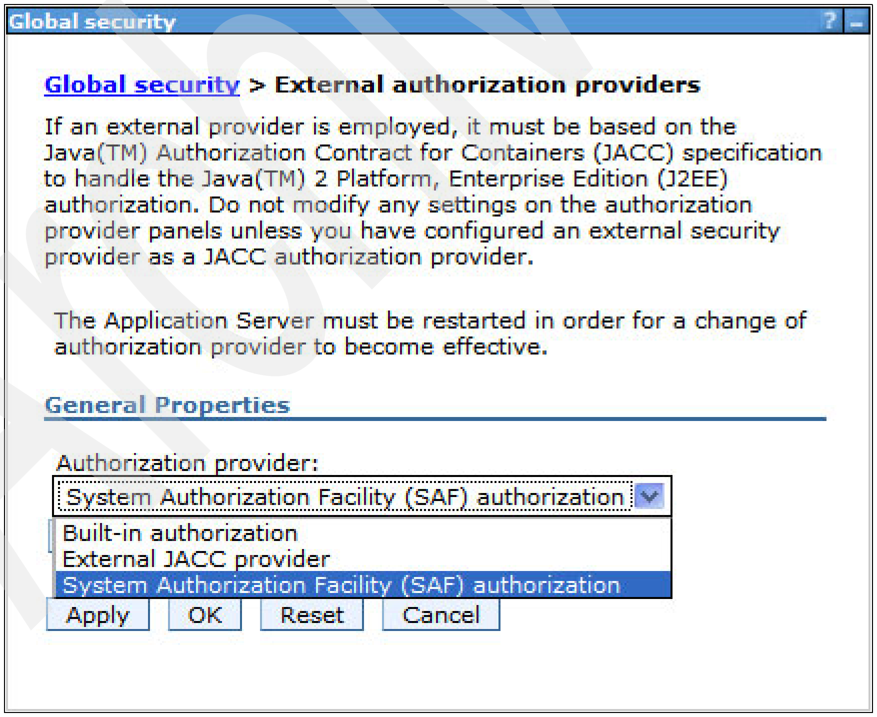
4. Select the authorization provider:

a. Click **Security** → **Global security** → **External authorization providers**.

> **Important:** The security configuration wizard turns off SAF authorization. If you run the security configuration wizard after having enabled SAF authorization, you need to manually turn on SAF authorization again.

b. Select either the SAF authorization or Built-in authorization as shown in Figure 11-4, and click **Apply**:

- If the built-in authorization (default authorization) is selected, SAF is used as the user registry, but it is not the repository for user-to-role mapping.
- If you choose SAF as your authorization provider, SAF is used for the user registry, as well as for the user-to-role mapping.



Global security > External authorization providers

If an external provider is employed, it must be based on the Java(TM) Authorization Contract for Containers (JACC) specification to handle the Java(TM) 2 Platform, Enterprise Edition (J2EE) authorization. Do not modify any settings on the authorization provider panels unless you have configured an external security provider as a JACC authorization provider.

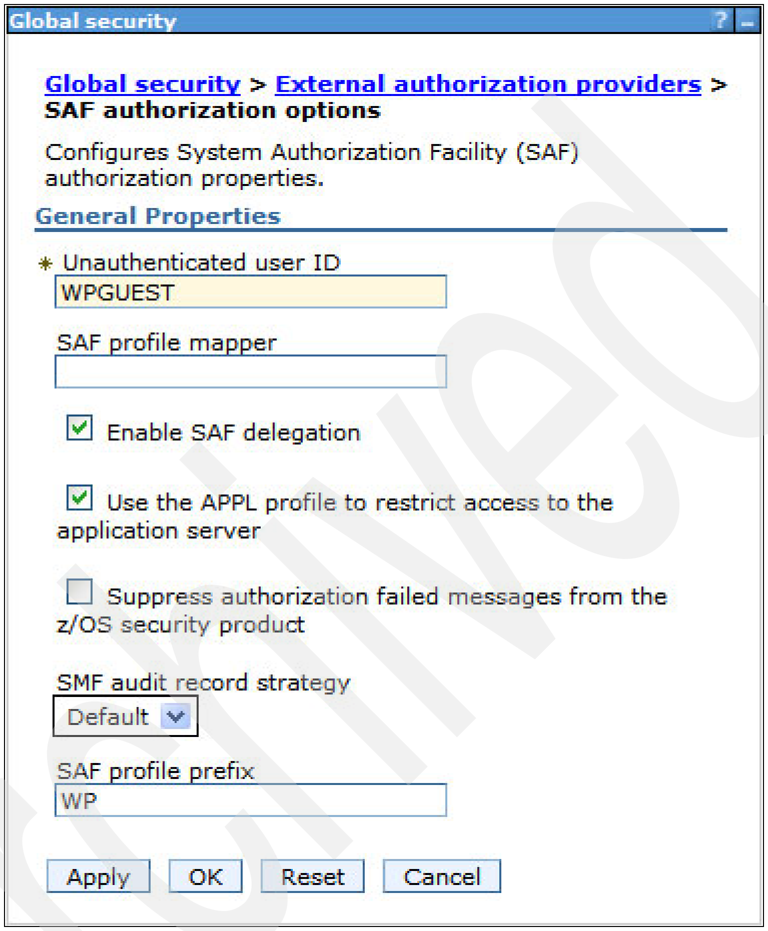
The Application Server must be restarted in order for a change of authorization provider to become effective.

General Properties

Authorization provider: System Authorization Facility (SAF) authorization

Figure 11-4 Step 4: External authorization providers

c. If you choose the SAF authorization provider, click **Configure** for the detailed settings that are shown in Figure 11-5.



Global security > External authorization providers > **SAF authorization options**

Configures System Authorization Facility (SAF) authorization properties.

General Properties

* Unauthenticated user ID: WPGUEST

SAF profile mapper:

[✓] Enable SAF delegation

[✓] Use the APPL profile to restrict access to the application server

[ ] Suppress authorization failed messages from the z/OS security product

SMF audit record strategy: Default

SAF profile prefix: WP

Figure 11-5 SAF authorization options

Specify the unauthenticated user ID. It is common practice to define the unauthenticated user ID with both the restricted and protected attributes.

The *SAF profile prefix*, formerly known as the SAF security domain, specifies a prefix that will be added to all EJBROLE profiles that are used for the Java EE roles. Moreover, this prefix is also used as the APPL profile name. If your environment consists of multiple cells within the one logical partition (LPAR) or one sysplex, this prefix can be used to distinguish the administrator role in each cell. A common approach is to use the two character abbreviation of the cell (in this case, WP) as the SAF profile prefix. As explained in 10.5.1, “SAF authorization for administrative roles" on page 393, the EJBROLE profile for the administrator is defined as shown in the following example, where the *<domain>* is the SAF profile prefix:

```
RDEFINE EJBROLE <domain>.administrator UACC(NONE)
```

In this example, WP is the SAF profile prefix:

```
RDEFINE EJBROLE WP.administrator UACC(NONE)
```

If the SAF profile prefix is not explicitly specified, no *<domain>* prefix is added to the EJBROLE profiles. The default value of CBS390 will be used as the APPL profile name.

If SAF authorization is enabled during the customization process, **Use the APPL profile to restrict access to the application server** is selected by default. The APPL profile will be selected prior to the EJBROLE profile during the process of authorization, if this option is enabled. Consequently, the access to the administrative console will be denied for a user that is mapped to the EJBROLE profile administrator, but not to the appropriate APPL profile. In the case where *<domain>* is the SAF profile prefix, if there is no prefix defined, CBS390 is the default profile.

5. Save and synchronize the changes with nodes.

> **Important:** At this point, it is very important for a Network Deployment environment to synchronize the changes with nodes. If this important security change is not distributed to the nodes, the node agents will start with *global security off* whereas the deployment manager will start with *global security on*. Finally, the components will not be able to communicate with each other. The only fix is described in 11.3, "Disabling administrative security" on page 428.

In a Network Deployment environment, the synchronization can be initiated by selecting **System administration** → **Save changes to master repository**, and click **Save**.

6. If the SAF authorization provider has been selected and global security with a z/OS security product has not been enabled during the customization process, you must define many additional SAF profiles, certificates, and keyrings.

Tip: In the case of RACF, the easiest way to get the required definitions is to regenerate the RACF jobs for each node by selecting "Use z/OS security product" in the Profile Management Tool (PMT).

Existing PMT profiles can be changed by using Regenerate. After customization, the generated RACF definitions can be uploaded to the host and will be stored in *hlq*.DATA. If the uploaded node is a deployment manager node, the RACF definitions will be named BBODBRAC, and for any other node, it will be named BBOWBRAC.

If you work with an existing environment that used the ISPF panels for customization in the past, you can import the SAVECFG dataset into the PMT to work with the existing profile customization.

The EJBROLES profiles need to be defined to SAF as explained in 10.5.1, "SAF authorization for administrative roles" on page 393.

7. Administrative security will be active after restarting the complete cell. When logging on to the administrative console, you will need to provide the primary administrative user ID and its password. Note that all security-related changes to the WebSphere configuration need a restart of the cell in order to be activated.

11.3 Disabling administrative security

If anything goes wrong during the configuration of administrative security, you might not be able to log on to the administrative console, or other security-related problems might occur. There are two ways to disable administrative security.

If you can still log on to the administrative console, you can disable security through the administrative console:

1. Click **Security** → **Global security**.
2. Clear the **Enable administrative security** check box.
3. Save the changes and synchronize changes with nodes.
4. Restart the server.

If you cannot log on to the administrative console, you can disable the administrative security using the **`wsadmin`** command-line interface:

1. Stop the server.
2. Log in to a UNIX System Services shell via OMVS or telnet.
3. Go to the `*profile_root*/bin` directory.

4. Enter the **wsadmin** shell by entering this command:

   ```
   wsadmin.sh -conntype NONE
   ```

   The connection type `NONE` does not require a running server process. In this mode, **wsadmin** will change the configuration in the appropriate XML configuration files and rebuild the `was.env` config file.

   If you have a Network Deployment environment, this command needs to be executed in the `bin` subdirectory of each node, because the nodes cannot be synchronized with this connection type.

   In the case of a certificate problem, the node agent will not be able to synchronize with the deployment manager. The communication between the deployment manager and node agent is secured through SSL with enabled administrative security.

5. Enter `securityoff`, and then exit wsadmin.
6. Start the server.

Example 11-1 shows an example of the **wsadmin** execution output. The `/wasconfig/wpcell/wpdmnode/DeploymentManager/profiles/default` directory is *`profile_root`* in this example.

Example 11-1 Command execution output of disabling security

```
SC04:/wasconfig/wpcell/wpdmnode/DeploymentManager/profiles/default/bin>
wsadmin.sh -conntype NONE
WASX7357I: By request, this scripting client is not connected to any
server process. Certain configuration and application operations will
be available in local mode.
WASX7029I: For help, enter: "$Help help"
wsadmin>securityoff
LOCAL OS security is off now but you need to restart server1 to make it
affected.

wsadmin>exit
```

11.4 Security trace

When you have administrative security enabled but administrative operations fail to execute, you must search for the cause of the problem. Setting a security trace helps you identify the security configuration problem. If the problems were introduced as a result of enabling security, it is safe to assume that a problem exists in the security configuration.

The security trace can be set in the administrative console. Select **Troubleshooting** → **Log and Trace** → *server_name* → **Change Log Detail Levels**.

You can change the trace level in the Configuration tab to enable the trace at server startup or in the Runtime tab to take effect immediately.

Figure 11-6 shows an example of the trace options and the trace levels.

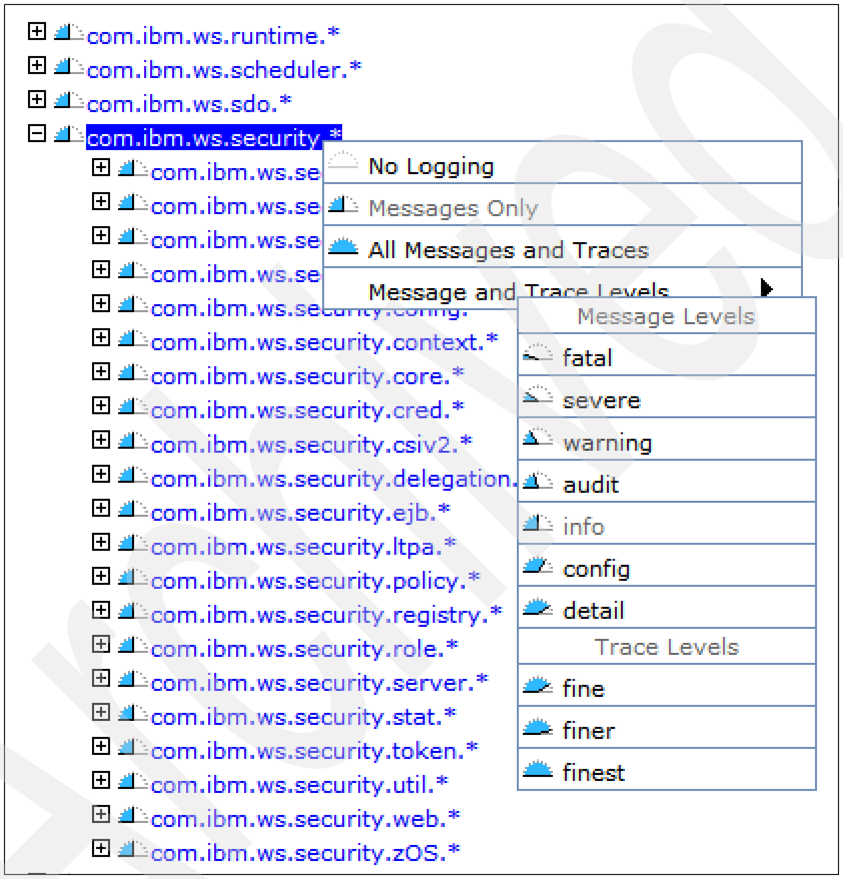

com.ibm.ws.runtime.*
com.ibm.ws.scheduler.*
com.ibm.ws.sdo.*
com.ibm.ws.security.*
com.ibm.ws.se
com.ibm.ws.se
com.ibm.ws.se
com.ibm.ws.se
com.ibm.ws.se
com.ibm.ws.security.context.*
com.ibm.ws.security.core.*
com.ibm.ws.security.cred.*
com.ibm.ws.security.csiv2.*
com.ibm.ws.security.delegation.
com.ibm.ws.security.ejb.*
com.ibm.ws.security.ltpa.*
com.ibm.ws.security.policy.*
com.ibm.ws.security.registry.*
com.ibm.ws.security.role.*
com.ibm.ws.security.server.*
com.ibm.ws.security.stat.*
com.ibm.ws.security.token.*
com.ibm.ws.security.util.*
com.ibm.ws.security.web.*
com.ibm.ws.security.zOS.*
No Logging
Messages Only
All Messages and Traces
Message and Trace Levels
Message Levels
fatal
severe
warning
audit
info
config
detail
Trace Levels
fine
finer
finest

Figure 11-6 Trace options and levels

An alternative method is to change the trace options/level using the MVS modify command:

```
F CR_short_name,tracejava='com.ibm.ws.security.*=all'
```

This command starts the trace dynamically and does not require a server restart.

After the traces are captured, you can use the following modify command to reset to the initial trace settings:

```
F CR_short_name,traceinit
```

If a trace of the deployment manager is required, use the modify commands in order to avoid additional trace overhead through the trace level modifications in the administrative console.

For more information about the available modify command options, refer to the WebSphere Information Center at:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/
com.ibm.websphere.zseries.doc/info/zseries/ae/rxml_mvsmodify.html
```

WebSphere z/OS and user registries

User registries or *repositories* are essential to security architectures. They contain critical user and group information. WebSphere Application Server for z/OS V7.0 provides flexible features to access various types of registries. The new federated repository functionality extends its capabilities even further.

This chapter describes the following topics:

- "Introduction to user registries" on page 434
- "Our scenario and our environment" on page 437
- "Stand-alone LDAP registry" on page 438
- "Federated repositories" on page 458

12.1 Introduction to user registries

In WebSphere Application Server for z/OS V7.0, the user registry or repository authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization. The information about users and groups resides in a registry or repository.

WebSphere Application Server for z/OS V7.0 provides implementations that support multiple types of registries and repositories:

- Local operating system
- Stand-alone Lightweight Directory Access Protocol (LDAP) registry
- Stand-alone custom registry
- Federated repositories

With WebSphere Application Server for z/OS V7.0, you use a user registry or repository for the following functions:

- Authenticating a user using basic authentication, identity assertion, client certificates, or tokens
- Retrieving information about users and groups to perform security-related administrative functions, such as mapping users and groups to security roles

Although the WebSphere Application Server supports multiple types of user registries, only one user registry can be active. This active registry is shared by all of the product server processes within a cell.

12.1.1 Local operating system registry

System Authorization Facility (SAF) interfaces are defined by z/OS to enable applications to use system authorization services or registries to control access to resources, such as data sets and z/OS commands. SAF allows security authorization requests to be processed directly through RACF or a third-party z/OS security provider.

Unlike the distributed platforms, the local operating system registry on the z/OS platform can be used in a multi-server z/OS environment, because the security database can be easily shared across the sysplex. In such a configuration, multiple WebSphere Application Server servers on separate z/OS logical partitions (LPARs) share the same security database.

The local operating system registry on z/OS is compatible with z/OS Enterprise Information Systems (EISs), such as CICS, IMS, and DB2. This compatibility is a major benefit when sending the original identity from WebSphere Application Server for z/OS V7.0 to the back-end EIS.

Use a local operating system registry in these situations:

- The authenticated users are present in the local security subsystem (intranet).
- The best performance available is mandatory.
- Comprehensive end-to-end security is needed (user ID existing in the Web server, Web container, EJB container, and back-end system).
- Good auditing support is needed.
- The users and application security need to be managed by the RACF security administrators.
- OS thread security or thread identity support is needed.

12.1.2 Stand-alone Lightweight Directory Access Protocol registry

Lightweight Directory Access Protocol (LDAP) is widely used in a distributed environment where multiple servers need access to a central user registry. LDAP is an option on z/OS as well.

This functionality allows a more versatile WebSphere environment, making room for cross-platform integration by allowing the use of existing user registries and authorization tables with the security functions in WebSphere Application Server.

LDAP servers act as a repository for user and group information. WebSphere Application Server for z/OS V7.0 binds to the LDAP server to retrieve this user and group information. This support is provided by using user and group *filters*. These filters are highly configurable to access all sorts of LDAP servers.

LDAP servers are incompatible with z/OS EISs, such as CICS, IMS, and DB2. More complex steps are required to access these systems with the user identity.

You must use an LDAP registry in the following situations:

- A single sign-on (SSO) solution with a distributed system is needed.
- A registry on other platforms must be used.
- A cross-platform authentication mechanism is mandatory.
- User identity must be maintained across multiple environments.
- RACF identities have to be used on distributed platforms through a central z/OS LDAP SDBM back end.
- RACF passwords need to be checked on distributed platforms through a central z/OS LDAP TDBM back end with native authentication.

12.1.3 Stand-alone custom registry

A *stand-alone custom registry* is a client-implemented registry that implements the UserRegistry Java interface that is provided by the product. A custom-implemented registry can support virtually any type of account repository from relational databases to flat files and so on. The custom user registry provides flexibility in adapting product security to various environments where a registry or repository other than federated repositories, an LDAP registry, or a local operating system registry already exists in the operational environment. It allows you to plug in any kind of registry whose support is not implemented by WebSphere Application Server security.

A custom registry is written as a Java program that implements a WebSphere Application Server for z/OS V7.0-supplied com.ibm.websphere.security.UserRegistry interface. Make sure that the implementation is not dependent on WebSphere Application Server resources (for example, data sources).

12.1.4 Federated repositories

Federated repositories enable you to use multiple repositories simultaneously. Federated repositories, which can include file-based repositories, LDAP repositories, or a sub-tree of an LDAP repository, are defined as being combined under a single realm. All of the user repositories that are configured under the federated repository functionality are transparent to WebSphere Application Server for z/OS V7.0.

The federated repositories functionality supports the logical joining of entries across multiple user repositories when the application server searches and retrieves entries from the repositories. For example, when an application calls for a sorted list of people whose age is greater than twenty, WebSphere Application Server for z/OS V7.0 searches all of the repositories in the federated repositories configuration. The results are combined and sorted before the application server returns the result to the application.

Unlike the local operating system, stand-alone LDAP registry, or custom registry options, federated repositories provide user and group management with read and write capabilities from the WebSphere administrative tools. If you do not configure the federated repositories functionality or do not enable federated repositories as the active repository, you cannot use the user management capabilities that are associated with federated repositories.

More information about federated repositories is provided in 12.4.1, “Federated repositories” on page 458.

12.2 Our scenario and our environment

This section focuses on configuring WebSphere Application Server for z/OS V7.0 with various registry types.We define an LDAP tree to be accessed by WebSphere Application Server for z/OS V7.0 in various manners, illustrating typical WebSphere configurations. This LDAP tree is shown in Figure 12-1.

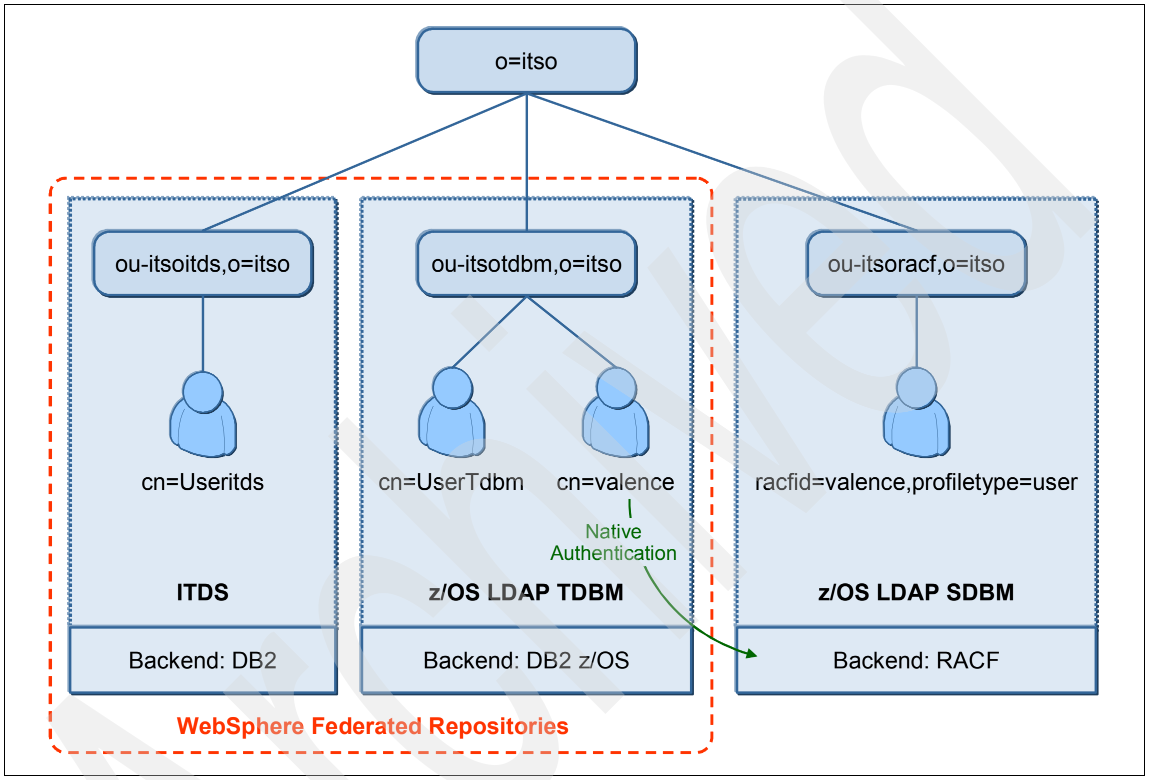


Figure 12-1 Our scenario LDAP tree and supporting LDAP servers

This LDAP tree common root organization is o=itso. This LDAP tree has three organizational units: `ou=itsoitds`, `ou=itsotdbm`, and `ou=itsoracf`:

- The `itsoitds` organizational unit is supported by a IBM Tivoli Directory Server V6. DB2 V8.2 on a Windows platform is used as a back end. The suffix of this subtree is `ou=itsoitds,o=itso`.

- The `itsotdbm` organizational unit is supported by a z/OS LDAP Technical Database Management (TDBM) server. DB2 z/OS V8 is used as a back end. This server runs on z/OS V1R9. The suffix of this subtree is `ou=itsotdbm,o=itso`. This server has one user configured for native authentication, which means that the password is checked against the RACF database.
- The `itsoracf` organizational unit is supported by a z/OS LDAP SDBM server. The RACF database is used as a back end. This server runs on z/OS V1R9. The suffix of this subtree is `ou=itsoracf,o=itso`.

Each of these organizational units have one or two users.

In this section, we describe how to configure WebSphere Application Server for z/OS V7.0 to use parts of this LDAP tree as a user registry. More specifically, we explain how to configure these registries:

- *Stand-alone* LDAP registry with WebSphere and z/OS LDAP SDBM back end (RACF)
- *Stand-alone* LDAP registry with WebSphere and z/OS LDAP TDBM back end (DB2)
- *Stand-alone* LDAP registry with WebSphere and z/OS LDAP TDBM native authentication
- *Federated repositories,* including Federated z/OS LDAP with TDBM back end (DB2) and Federated IBM Tivoli Directory Server

12.3 Stand-alone LDAP registry

The stand-alone LDAP registry feature in WebSphere Application Server for z/OS V7.0 is similar to the LDAP registry feature in V6 and V6.1.

In this section, we show how to configure the WebSphere stand-alone LDAP registry with z/OS LDAP SDBM, with z/OS LDAP TDBM with native authentication, and with IBM Tivoli Directory Server.

12.3.1 WebSphere and z/OS LDAP SDBM back end (RACF)

z/OS LDAP can handle many types of back ends. One of them is SDBM, which uses the RACF database as a data repository. With an SDBM back end, RACF user profiles, group profiles, and user to group connections appear as LDAP entries with a distinguished name to LDAP clients. All binds are authenticated with a RACF distinguished name and a RACF password.

z/OS LDAP SDBM supports add, modify, delete, and search operations. The access controls for user and group profiles are the RACF privileges of the authenticated user.

z/OS LDAP SDBM configuration

The z/OS LDAP installation is described in *Distributed Security and High Availability with Tivoli Access Manager and WebSphere Application Server for z/OS*, SG24-6760. In this section, we focus on the configuration.

To configure z/OS LDAP with an SDBM back end:

1. Find the LDAP configuration in the SLAPDCNF member of the LDAP customization data set. In our environment, the z/OS LDAP configuration file is `WTSC58.LDAP1.CNTL(SLAPDCNF)`. In this LDAP configuration file, remove the comment or set up the following parameters:

   ```
   database sdbm GLDBSDBM
   suffix "ou=itsoracf,o=itso"
   ```

   The suffix is the top of the LDAP tree that you want for your organization. We choose `ou=itsoracf,o=itso` in our environment. The suffix does not necessarily need to have an organizational unit (ou=). It might contain only an organization (o=).

   Example 12-1 shows an extract of our z/OS LDAP configuration for an SDBM back end.

   Example 12-1 z/OS LDAP configuration with an SDBM back end

   ```
   listen ldap://:3389
   maxConnections 60
   adminDN "cn=LDAP Administrator"
   adminPW "secret"
   #  SDBM-specific CONFIGURATION SETTINGS
   database sdbm GLDBSDBM
   suffix "ou=itsoracf,o=itso"
   ```

2. Restart the z/OS LDAP server. If your LDAP server is configured properly with an SDBM back end, a message similar to Example 12-2 shows in the LDAP log at startup.

   Example 12-2 z/OS LDAP log with an SDBM back end

   ```
   Backend type: sdbm, Backend ID: SDBM BACKEND
   SDBM BACKEND manages the following suffixes:
   Backend suffix: OU=ITSORACF,O=ITSO
   End of suffixes managed by SDBM BACKEND.
   Capability: LDAP_Backend_ID    Value: SDBM BACKEND
   ```

```
Capability: LDAP_Backend_BldDateTime     Value:
2006-04-18-15.18.14.000000
Capability: LDAP_Backend_APARLevel    Value: OA15948
Capability: LDAP_Backend_Release    Value: R 6.0
Capability: LDAP_Backend_Version    Value: V 1.0
Capability: LDAP_Backend_Dialect    Value: DIALECT 1.0
Capability: LDAP_Backend_BerDecoding    Value: STRING
Capability: LDAP_Backend_ExtGroupSearch    Value: YES
Capability: LDAP_Backend_krbIdentityMap    Value: YES
Capability: supportedControl   Value: 2.16.840.1.113730.3.4.2
Capability: supportedControl    Value: 1.3.18.0.2.10.2
End of capability listing for Backend type: sdbm, Backend ID: SDBM
BACKEND.
Backend capability listing ended.
Configuration file successfully read.
```

3. Validate that the SDBM is functional by accessing it from an LDAP client.

   Independently developed LDAP browser clients are available on the Web that you can use if you do not have a client.

   The LDAP client uses the following values for a connection in this example:

   ```
   Host : wtsc58.itso.ibm.com
   Port : 3389
   Base DN : ou=itsoracf,o=itso
   User DN : racfid=valence,profiletype=user,ou=itsoracf,o=itso
   ```

   The SDBM schema requires the RACF user distinguished name to follow this template:

   `racfid=`*`<racf_id>`*`,profiletype=user,`*`<sdbm_suffix>`*

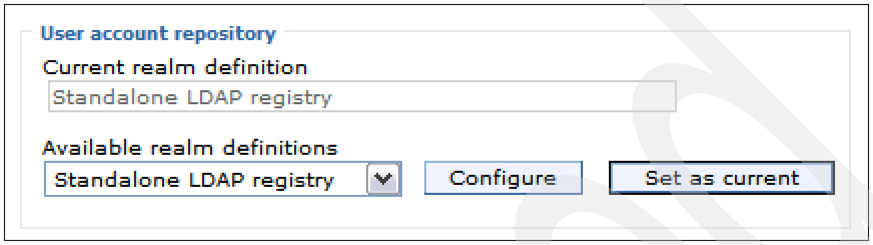
   The RACF user ID (`valence` in our example) must have the proper access level to list users and groups from the RACF database. It must be a RACF user ID with the AUDITOR attribute, a valid OMVS segment (specific or implied by a defaulted segment). It does not need a TSO segment. Using the client, you can access the RACF content displayed as an LDAP tree. All the RACF user IDs and groups can be accessed.

WebSphere z/OS configuration for z/OS LDAP SDBM

WebSphere Application Server for z/OS V7.0 supports accessing z/OS LDAP with an SDBM back end (RACF) when configured as a stand-alone LDAP registry.

In this section, we explain how to configure WebSphere to access z/OS LDAP SDBM:

1. In the administrative console, select **Security** → **Global security**. Under User account repository, in the Available realm definitions list box, select **Standalone LDAP registry**, and then, click **Set as current**. Standalone LDAP registry then appears in the Current realm definition field (Figure 12-2).



Current realm definition: Standalone LDAP registry
Available realm definitions: Standalone LDAP registry

Figure 12-2 WebSphere user account repository

   Verify that administrative security is enabled and also that application security is enabled if necessary. Click **Apply** on this page to keep these settings.

2. Click **Configure** and set up the general properties for the z/OS LDAP server (Figure 12-3 on page 442):

   – Primary administrative user name: This ID is the security server ID, which is only used for WebSphere Application Server administrative security and is not associated with the system process that runs the server. We use `racfid=valence,profiletype=user,ou=itsoracf,o=itso` in our environment. This primary user name will be used to log on to the administrative console, for example.

   – Server user identity: Select **Automatically generated server identity** in a WebSphere 6.1 and V7.0 environment.

   – Type of LDAP server: Select **Custom**.

   – Host and port: Enter the full Domain Name System (DNS) host name and TCP port to access z/OS LDAP. We use `wtsc58.itso.ibm.com` and `3389` in our environment.

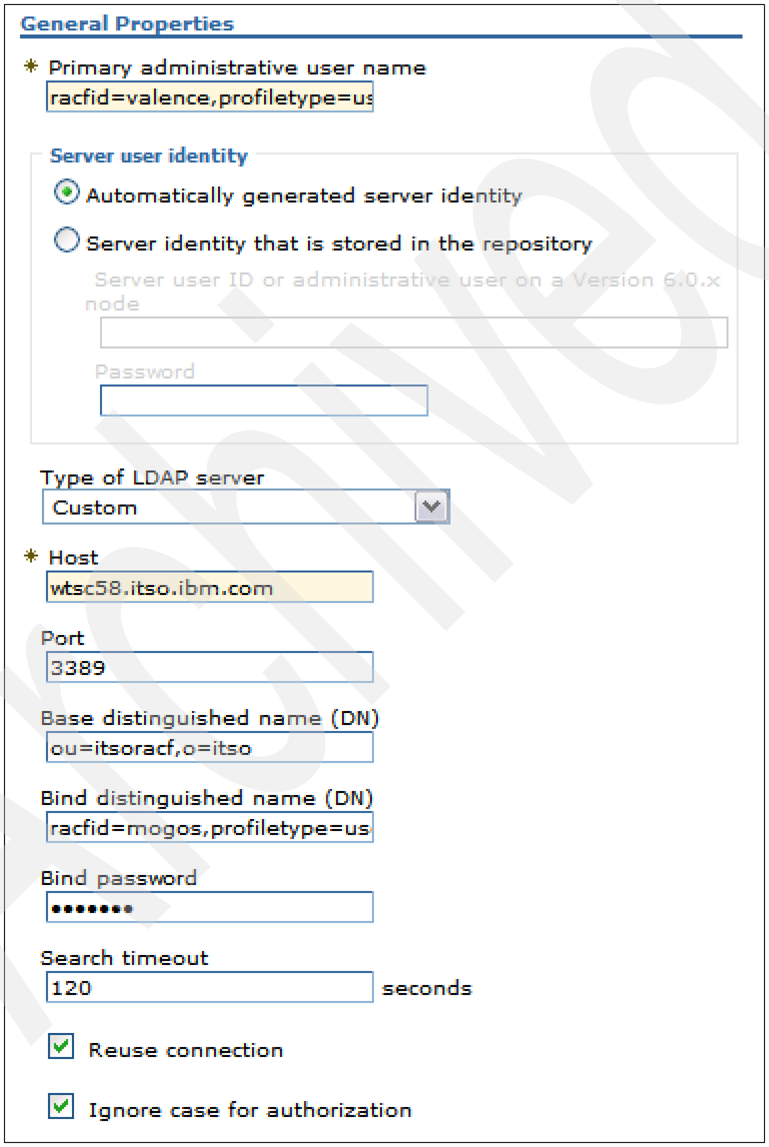
   – Base distinguished name (DN): The base DN indicates the starting point for searches in this LDAP directory server. It is the suffix `ou=itsoracf,o=itso` in our environment.

   – Bind distinguished name (DN) and password: The bind DN is required if anonymous binds are not possible on the LDAP server to obtain user and group information.

We use `racfid=mogos,profiletype=user,ou=itsoracf,o=itso` in our environment. This ID is the user ID to connect to LDAP. It is allowed to list RACF users and groups.

– Make sure that **Ignore case for authorization** is selected. RACF user names and group names are not case-sensitive.

Then, click **Apply**.



General Properties

* Primary administrative user name: racfid=valence,profiletype=us

Server user identity

(●) Automatically generated server identity

( ) Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

Type of LDAP server: Custom

* Host: wtsc58.itso.ibm.com

Port: 3389

Base distinguished name (DN): ou=itsoracf,o=itso

Bind distinguished name (DN): racfid=mogos,profiletype=us

Bind password: •••••••

Search timeout: 120 seconds

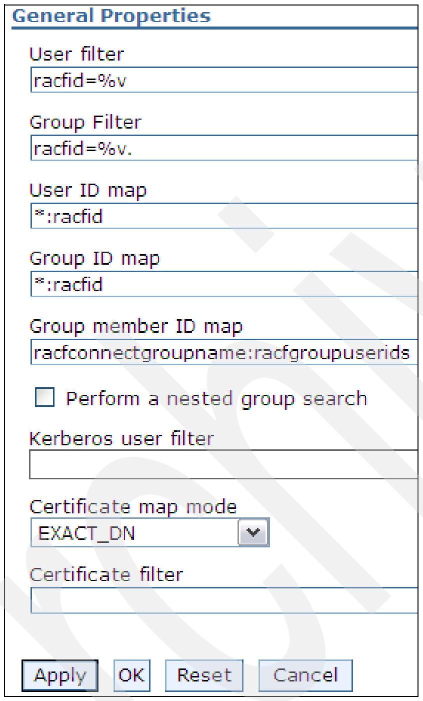
[✓] Reuse connection

[✓] Ignore case for authorization

Figure 12-3 WebSphere configuration for z/OS LDAP SDBM

3. In the same window, under Additional Properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry setting** and configure the properties (Figure 12-4):
   - Change user filter and group filter to `racfid=%v`.
   - Change user ID map and group ID map to `*:racfid`.
   - Change group member ID map to `racfconnectgroupname:racfgroupuserids`.

   Click **OK**.


General Properties

User filter
racfid=%v

Group Filter
racfid=%v.

User ID map
*:racfid

Group ID map
*:racfid

Group member ID map
racfconnectgroupname:racfgroupuserids

☐ Perform a nested group search

Kerberos user filter

Certificate map mode
EXACT_DN

Certificate filter

Figure 12-4 WebSphere advanced configuration for z/OS LDAP SDBM

4. Save the changes in the master repository, and restart the cell.

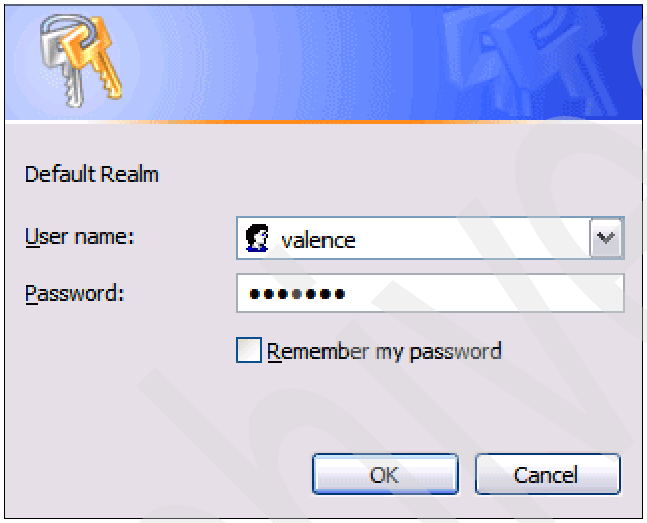
WebSphere z/OS LDAP SDBM back-end validation

After restarting WebSphere Application Server for z/OS V7.0, log in to the administrative console with the primary administrative user name that was defined earlier. You can use the full distinguished name (for our example, (`racfid=valence,profiletype=USER,ou=itsoracf,o=itso`) or the user name only (`valence`). The user name is a RACF identity.

You can also validate the user registry with the snoop servlet, which is bundled in WebSphere for z/OS. With application security enabled, the snoop servlet requires basic authentication. Call the snoop with a URL, such as:

```
http://wtsc58.itso.ibm.com:49080/snoop/
```

Authenticate providing a z/OS LDAP SDBM user name and password (RACF user name and password) as shown in Figure 12-5. The snoop servlet then appears and shows the authenticated principal.


Default Realm

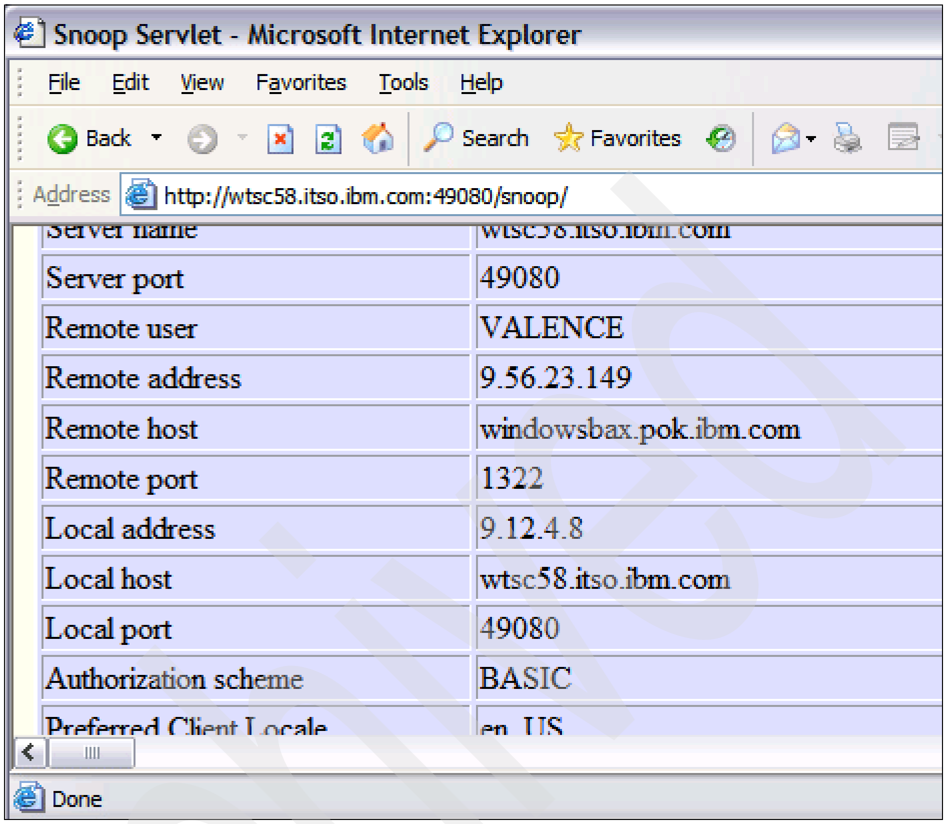
User name: valence

Password: ●●●●●●●

☐ Remember my password

Figure 12-5 Snoop servlet authentication with a RACF user ID

When authenticated, the snoop servlet displays information, including the authenticated user (`valence`, in our example). Refer to Figure 12-6 on page 445.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://wtsc58.itso.ibm.com:49080/snoop/
Done

Server name	wtsc58.itso.ibm.com
Server port	49080
Remote user	VALENCE
Remote address	9.56.23.149
Remote host	windowsbax.pok.ibm.com
Remote port	1322
Local address	9.12.4.8
Local host	wtsc58.itso.ibm.com
Local port	49080
Authorization scheme	BASIC
Preferred Client Locale	en_US

Figure 12-6 Snoop servlet showing authenticated RACF user ID

WebSphere, LDAP SDBM, and SAF authorization

When LDAP is the configured user registry, it is common to bind users and groups to Java EE roles at application deployment time. Bindings to administrative and naming roles are usually done through the administrative console. These bindings are referred to as *WebSphere bindings*. In the case of LDAP SDBM, all users and groups need to be existing RACF identities.

z/OS has a strong tradition in security and security administration. RACF administrators might not agree that a deployer (or development team if users and groups are already mapped to Java EE roles in the applications' descriptors) has the authority to decide which RACF group or user ID can be authorized to which Java EE role. The mapping of users and groups to roles at deployment time, or searching for users and groups in the RACF database, is done under the credentials of the bind distinguished name (BDN). This user ID must have the AUDITOR attribute. Therefore, audit trails only lead to that BDN identity, not to the identity of the deployer.

This section describes how to set up SAF authorization as an alternative to WebSphere bindings in combination with LDAP SDBM as the configured user registry.

Only a few steps are required to configure SAF authorization:

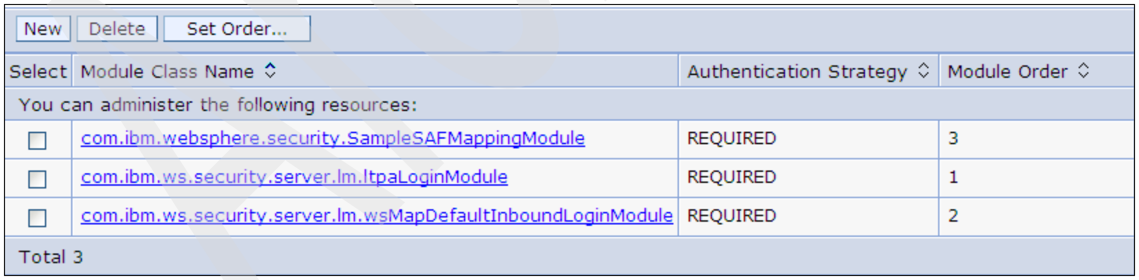
1. To configure WebSphere to use SAF authorization, select **Security** → **Global Security** → **Custom properties**. Set the value of `com.ibm.security.SAF.authorization` to true.

   A key component in configuring SAF authorization in combination with LDAP SDBM authentication is a mapping module. This module maps the LDAP distinguished name to a SAF identity. You do not need to write a JAAS module. The SampleSAFMappingModule module is provided in the WebSphere run time and functions in combination with LDAP SDBM-provided credentials. In an LDAP SDBM-constructed DN, the SAF ID is already part of the DN. We assume that Lightweight Third Party Authentication (LTPA) is the authentication mechanism. SWAM was deprecated after Version 6.1.

2. Add the SampleSAFMappingModule module to the JAAS login. Select **Security** → **Global Security.** Under Authentication, expand **Java Authentication and Authorization Service**, and click **System Logins**.

3. Modify DEFAULT, RMI_INBOUND, and WEB_INBOUND. The modification is identical for all three system login configurations. *Make sure that you alter all three of them.*

4. Click **Alias** in **JAAS login modules**, and click **New**. Enter `com.ibm.websphere.security.SampleSAFMappingModule` for the module class name, and select **Use login module proxy**. Verify that the authentication strategy is set to REQUIRED. Click **OK**. Figure 12-7 shows the search order.



Select	Module Class Name	Authentication Strategy	Module Order
You can administer the following resources:			
☐	com.ibm.websphere.security.SampleSAFMappingModule	REQUIRED	3
☐	com.ibm.ws.security.server.lm.ltpaLoginModule	REQUIRED	1
☐	com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule	REQUIRED	2
Total 3			

Figure 12-7 JAAS login modules on a system login alias

5. Make sure that the EJBROLE class is active and shows RACLIST. Optionally, configure the grouping class GEJBROLE. Define all administrative and naming roles and permit the appropriate user IDs and groups to the profiles. If you already have applications deployed, examine the Java EE roles and

define them as well in the EJBROLE class, with the appropriate access list. Both administrative and application security must be enabled. Remove all WebSphere bindings from the configuration. If you are migrating from WebSphere bindings to SAF authorization, define all EJBROLE class profiles and permits in advance.

6. Now, stop the cell or base server, and restart.

WebSphere differs from SAF bindings in that more than one application can specify the same Java EE role name. In the case of SAF bindings, authorization checks to the roles in those applications result in a SAF call to the same profile in the EJBROLE class, and therefore, grant access to all applications with identical role names. In the case of WebSphere bindings, the mapping is at the application level. We always recommend that the developers discuss role naming with the RACF security administrators.

12.3.2 WebSphere and z/OS LDAP TDBM back end (DB2)

This section provides information about configuring WebSphere for z/OS LDAP TDBM, which uses DB2 z/OS as a data repository.

z/OS LDAP TDBM configuration

The z/OS LDAP installation is described in *Distributed Security and High Availability with Tivoli Access Manager and WebSphere Application Server for z/OS*, SG24-6760. In this section, we focus on the configuration activities.

To configure z/OS LDAP with an TDBM back end:

1. Find the LDAP configuration in the SLAPDCNF member of the LDAP customization data set. In our environment, the z/OS LDAP configuration file is `WTSC58.LDAP1.CNTL(SLAPDCNF)`. In this LDAP configuration file, remove the comment or set up the following parameters:

   ```
   database tdbm GLDBTDBM
   suffix "ou=itsotdbm,o=itso"
   servername DB2B
   dbuserid GLDSRV
   databasename GLDDB
   dsnaoini WTSC58.LDAP1.CNTL(DSNAOINI)
   attroverflowsize 255
   ```

   The suffix is the top of the LDAP tree that you want for your organization. We choose `ou=itsotdbm,o=itso` in our environment. The suffix does not necessarily need to have an organizational unit (ou=). It can contain an organization only (o=). The DB2 back-end configuration refers to the DB2 setup that was done at z/OS LDAP installation time.

Example 12-3 shows an extract of our z/OS LDAP configuration to use a TDBM back end.

Example 12-3 z/OS LDAP configuration with a TDBM back end

```
listen ldap://:3389
maxConnections 60
adminDN "cn=LDAP Administrator"
adminPW "secret"
#  TDBM-specific CONFIGURATION SETTINGS
database tdbm GLDBTDBM
suffix "ou=itsotdbm,o=itso"
servername DB2B
dbuserid GLDSRV
databasename GLDDB
dsnaoini WTSC58.LDAP1.CNTL(DSNAOINI)
attroverflowsize 255
```

2. Restart the z/OS LDAP server. If your LDAP server is configured properly with a TDBM back end, a message similar to Example 12-4 appears in the LDAP log at startup.

Example 12-4 z/OS LDAP log with a TDBM back end

```
Backend type: tdbm, Backend ID: TDBM BACKEND
TDBM BACKEND manages the following suffixes:
Backend suffix: OU=ITSOTDBM,O=ITSO
End of suffixes managed by TDBM BACKEND.
Capability: LDAP_Backend_ID    Value: TDBM BACKEND
Capability: LDAP_Backend_BldDateTime     Value:
2006-07-25-22.56.16.000000
Capability: LDAP_Backend_APARLevel    Value: OA17138
Capability: LDAP_Backend_Release    Value: R 6.0
Capability: LDAP_Backend_Version    Value: V 1.0
Capability: LDAP_Backend_Dialect    Value: DIALECT 1.0
Capability: LDAP_Backend_BerDecoding    Value: BINARY
Capability: LDAP_Backend_ExtGroupSearch    Value: YES
Capability: LDAP_Backend_krbIdentityMap    Value: YES
Capability: supportedControl    Value: 2.16.840.1.113730.3.4.2
Capability: supportedControl    Value: 1.3.18.0.2.10.2
...
Capability: LDAP_Backend_SupportedCapabilities Value:
1.3.18.0.2.32.3
Capability: LDAP_Backend_SupportedCapabilities Value:
1.3.18.0.2.32.31
...
```

```
Capability: LDAP_Backend_EnabledCapabilities Value: 1.3.18.0.2.32.31
End of capability listing for Backend type: tdbm, Backend ID: TDBM
BACKEND.
```

3. Copy the following files to the LDAP working directory `/etc/ldap`:
   - `/usr/lpp/ldap/etc/schema.user.ldif`
   - `/usr/lpp/ldap/etc/schema.IBM.ldif`
4. Edit these files and change the line `cn=schema,<suffix>` to reflect the TDBM suffix that is defined in the z/OS LDAP configuration file, for example:

   ```
   dn: cn=schema,ou=itsotdbm,o=itso
   ```

   Important: There are no spaces between the comma (,) and o=. Those schema files contain the objects and attributes that are used to organize the data following the IBM schema and for the SAF native authentication object class.

5. From UNIX System Services, use the **`ldapmodify`** command to load the schema files into z/OS LDAP:

   ```
   ldapmodify -h wtsc58 -p 3389 -D "cn=LDAP Administrator" -w secret -f
   /etc/ldap/schema.user.ldif
   ldapmodify -h wtsc58 -p 3389 -D "cn=LDAP Administrator" -w secret -f
   /etc/ldap/schema.IBM.ldif
   ```

   Load `schema.user.ldif` followed by `schema.IBM.ldif`. The options here are:
   - `-h` *host name* defines the host name where LDAP is running.
   - `-p` *port number* defines the port on which LDAP is listening.
   - `-D` *adminDN* defines the administrator distinguished name (DN).
   - `-w` *password* is the administrator password.
6. Add a suffix entry and a person entry to the z/OS LDAP. For this example, create a new `schema.suffix.ldif` that contains the following information:

   ```
   dn: ou=itsotdbm,o=itso
   objectclass: organizationalUnit
   objectclass:top
   ou: itsotdbm

   dn: cn=UserTdbm,ou=itsotdbm,o=itso
   objectClass: inetOrgPerson
   objectClass: ePerson
   objectClass: organizationalPerson
   objectClass: person
   objectClass: top
   cn: UserTdbm
   ```

```
uid: UserTdbm
sn: 2006
description: Test user for TDBM
```

Customize these entries to match your suffix and the user name that you want for a first user.

Use a command similar to the following command to add the entries to the LDAP tree:

```
ldapadd -h wtsc58 -p 3389 -D "cn=LDAP Administrator" -w secret -f
schema.suffix.ldif
```

7. Set up a password for the new user by creating a file called `user.password.ldif`, which contains the following information:

```
dn: cn=UserTdbm,ou=itsotdbm,o=itso
changetype:modify
replace:userpassword
userpassword: usertdbm
```

Use a command similar to the following command to modify the user entry in the LDAP tree:

```
ldapmodify -h wtsc58 -p 3389 -D "cn=LDAP Administrator" -w secret -f
user.password.ldif
```

WebSphere z/OS configuration for z/OS LDAP TDBM

WebSphere Application Server for z/OS V7.0 supports accessing z/OS LDAP with a TDBM back end (DB2) when configured as a stand-alone LDAP registry. In this section, we explain how to configure WebSphere Application Server for z/OS V7.0 in order to access z/OS LDAP TDBM:

1. In the administrative console, select **Security** → **Global security**. Under User account repository, in the Available realm definitions list box, select **Standalone LDAP registry**, and then, click **Set as current**. The Standalone LDAP registry then appears in the Current realm definition field (Figure 12-8).

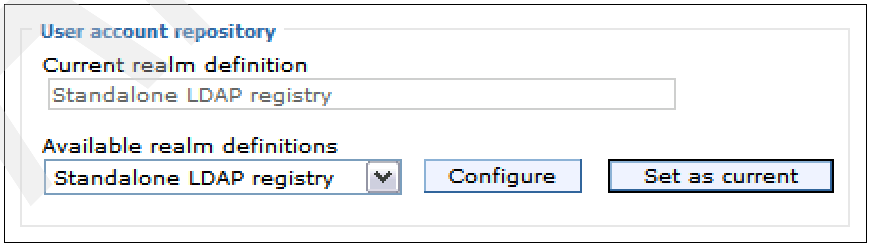
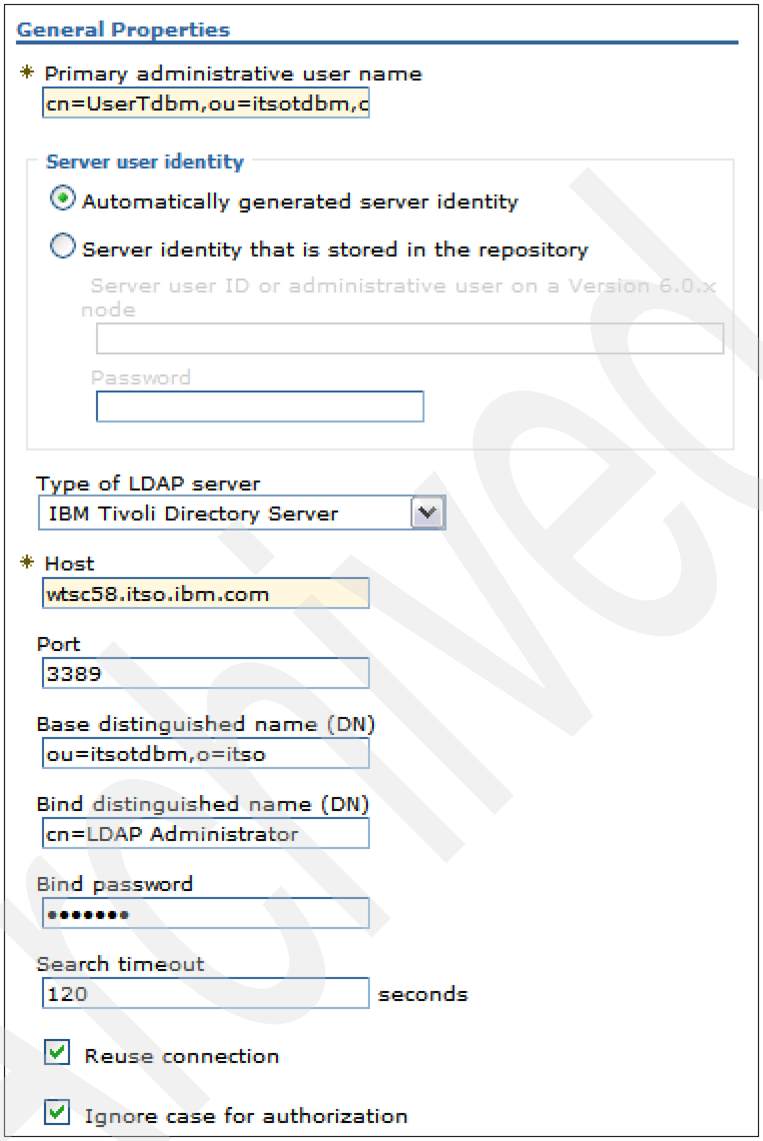


Figure 12-8 WebSphere user account repository

Verify that administrative security is enabled and, if required, application security. Click **Apply** on this page to keep these settings.

2. Click **Configure** (Figure 12-8 on page 450).
3. Set up the general properties for the z/OS LDAP server (Figure 12-9 on page 452):
   - Primary administrative user name: This ID is the security server ID, which is only used for WebSphere Application Server security and is not associated with the system process that runs the server. We use `cn=UserTdbm,ou=itsotdbm,o=itso` in our environment. This primary user name is used to log on to the administrative console, for example.
   - Leave **Automatically generated server identity** selected.
   - Type of LDAP server: Select **IBM Tivoli Directory Server**. This choice sets up the default filters for retrieving users and groups in the Advanced LDAP user registry settings.
   - Host and port: Enter the full DNS host name and TCP port to access z/OS LDAP. We use `wtsc58.itso.ibm.com` and `3389` in our environment.
   - Base distinguished name (DN): The base DN indicates the starting point for searches in this LDAP directory server. It is `ou=itsotdbm,o=itso` in our environment.
   - Bind distinguished name (DN) and password: The bind DN is required if anonymous binds are not possible on the LDAP server to obtain user and group information. We use `cn=LDAP Administrator` in our environment.

   Then, click **Apply**.


General Properties

* Primary administrative user name
cn=UserTdbm,ou=itsotdbm,c

Server user identity

[✓ Automatically generated server identity]

Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

Type of LDAP server
IBM Tivoli Directory Server

* Host
wtsc58.itso.ibm.com

Port
3389

Base distinguished name (DN)
ou=itsotdbm,o=itso

Bind distinguished name (DN)
cn=LDAP Administrator

Bind password
•••••••

Search timeout
120 seconds

[✓] Reuse connection

[✓] Ignore case for authorization

Figure 12-9 WebSphere configuration for z/OS LDAP TDBM

4. Save the changes in the master repository, and restart the cell.

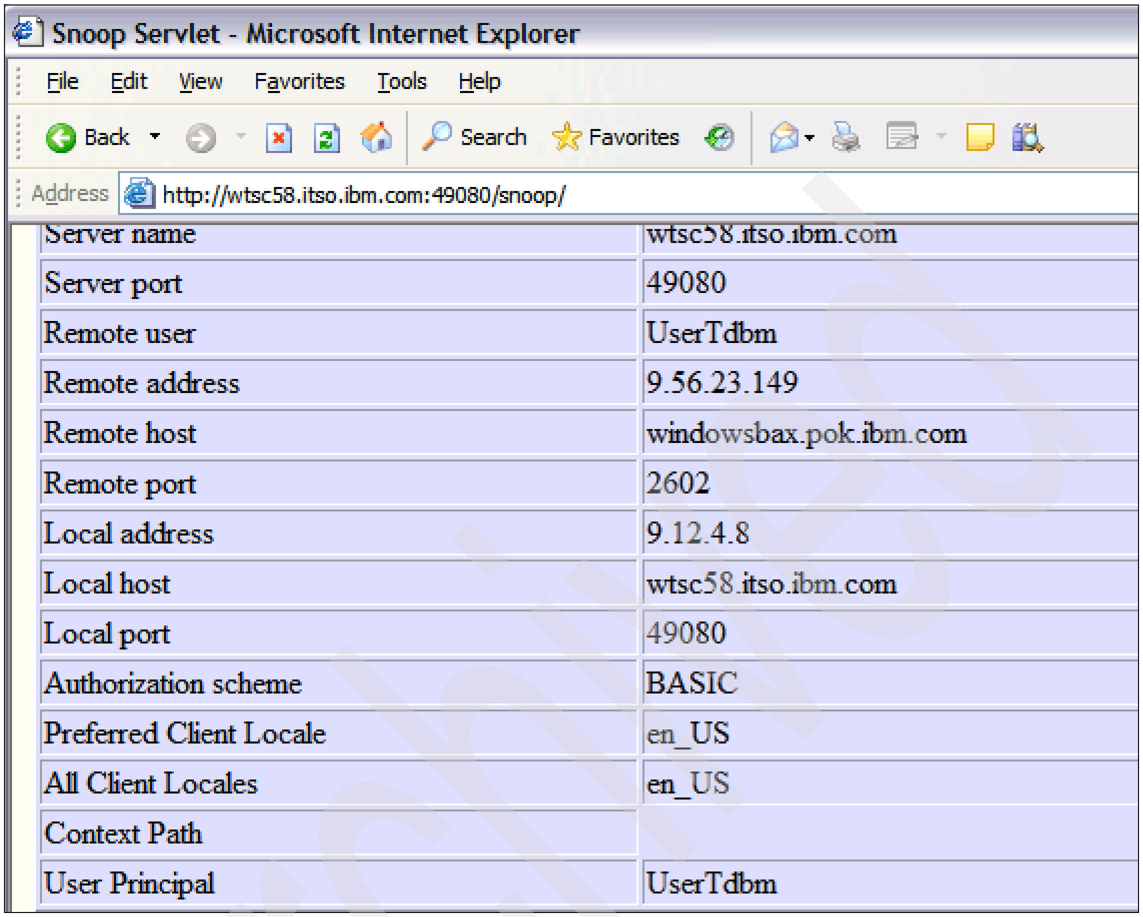
WebSphere and z/OS LDAP TDBM back-end validation

After restarting WebSphere Application Server for z/OS V7.0, log in to the administrative console with the primary administrative user name that was defined earlier. It is possible to use the full distinguished name (`cn=UserTdbm,ou=itsotdbm,o=itso`) or the user name only (`usertdbm`). Using our `ldif` file, the password is `usertdbm` also.

You can validate the user registry with the snoop servlet, which is bundled in WebSphere for z/OS. With the application security enabled, the snoop servlet requires basic authentication. Call the snoop servlet with a URL, such as:

```
http://wtsc58.itso.ibm.com:49080/snoop/
```

Authenticate by providing the user name and password that were defined earlier. The snoop servlet then appears and shows the authenticated principal. When authenticated, the snoop servlet displays information, including the authenticated user (`usertdbm`, in our example). Refer to Figure 12-10 on page 454.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wtsc58.itso.ibm.com:49080/snoop/

Server name	wtsc58.itso.ibm.com
Server port	49080
Remote user	UserTdbm
Remote address	9.56.23.149
Remote host	windowsbax.pok.ibm.com
Remote port	2602
Local address	9.12.4.8
Local host	wtsc58.itso.ibm.com
Local port	49080
Authorization scheme	BASIC
Preferred Client Locale	en_US
All Client Locales	en_US
Context Path	
User Principal	UserTdbm

Figure 12-10 Snoop servlet showing authenticated z/OS LDAP TDBM user ID

12.3.3 WebSphere and z/OS LDAP TDBM native authentication

LDAP has the ability to authenticate to RACF through TDBM by supplying a RACF password on a simple bind to a TDBM back end. Authorization information is still gathered by the LDAP server based on the distinguished name that performed the bind operation. The LDAP entry that contains the bind DN must contain either the `ibm-nativeId` attribute or `uid` attribute to specify the ID that is associated with this entry. Note that the SDBM back end does not have to be configured. The ID and password are passed to RACF, and the verification of the password is performed by RACF. Another feature of native authentication is the ability to change your RACF password by issuing an LDAP modify command.

You need to enable native authentication for the following reasons:

- You have the need for a central user registry with RACF identities (SSO).
- You want the ability to reuse RACF user IDs and passwords using an LDAP interface.
- You plan to use a security product, such as Tivoli Access Manager, as a front end to WebSphere Application Server for z/OS V7.0.

z/OS LDAP TDBM native authentication configuration

First, configure LDAP z/OS with a TDBM back end, as described in 12.3.2, “WebSphere and z/OS LDAP TDBM back end (DB2)” on page 447.

Additional modification is needed in the LDAP configuration file. Specify the native authentication options in this configuration file in the TDBM section. To do so, remove the comment from the following directives:

- `useNativeAuth`: This line defines which attribute uses native authentication. We use the `selected` value, which means that only the `ibm-nativeId` attribute is subject to native authentication.
- `nativeUpdateAllowed`: This line defines whether LDAP can modify attributes, such as the password, for the native authentication system. We choose `on`.
- `nativeAuthSubtree`: This line defines in which subtree in the LDAP tree native authentication occurs. This option can appear multiple times to specify all subtrees that use native authentication. If this option is omitted or is set to **all**, the entire directory is subject to native authentication. If **useNativeAuth selected** or **all** is not specified, this option is ignored.

  Example 12-5 shows an extract of our z/OS LDAP configuration to use native authentication with a TDBM back end.

Example 12-5 z/OS LDAP configuration for native authentication

```
listen ldap://:3389
maxConnections 60
adminDN "cn=LDAP Administrator"
adminPW "secret"
#  TDBM-specific CONFIGURATION SETTINGS
database tdbm GLDBTDBM
suffix "ou=itsotdbm,o=itso"
servername DB2B
dbuserid GLDSRV
databasename GLDDB
dsnaoini WTSC58.LDAP1.CNTL(DSNAOINI)
attroverflowsize 255
useNativeAuth selected
```

```
nativeUpdateAllowed on
```

Restart the LDAP server to activate these configuration modifications. You now see the following message in the LDAP JOBLOG:

```
The useNativeAuth configuration option SELECTED has been enabled.
```

When using the TDBM back end for native authentication, users need to have the ibm-nativeAuthentication objectclass and ibm-nativeId attribute. If you have existing users in your LDAP TDBM back end, you need to modify their definitions to include the ibm-nativeAuthentication objectclass and ibm-nativeId attribute.

For our configuration, we create a new user with the ibm-nativeAuthentication objectclass and ibm-nativeId attribute using a file called `newuser.ldif`, which contains the following information:

```
dn: cn=valence,ou=itsotdbm,o=itso
objectClass: inetOrgPerson
objectClass: ePerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: valence
uid: valence
sn: 2006
description: Test user for TDBM Native
ibm-nativeId: VALENCE
objectclass: ibm-nativeAuthentication
```

Use a command similar to the following command to add the entries to the LDAP tree:

```
ldapadd -h wtsc58 -p 3389 -D "cn=LDAP Administrator" -w secret -f
newuser.ldif
```

The `ldif` entry does not have a password. The password will be verified against the `VALENCE` entry in the RACF database.

The `ibm-nativeId` attribute specifies the user ID to which the entry binds in the RACF database. In this example, the `valence` LDAP entry binds to the user `VALENCE` in the RACF database.

WebSphere z/OS configuration for LDAP native authentication

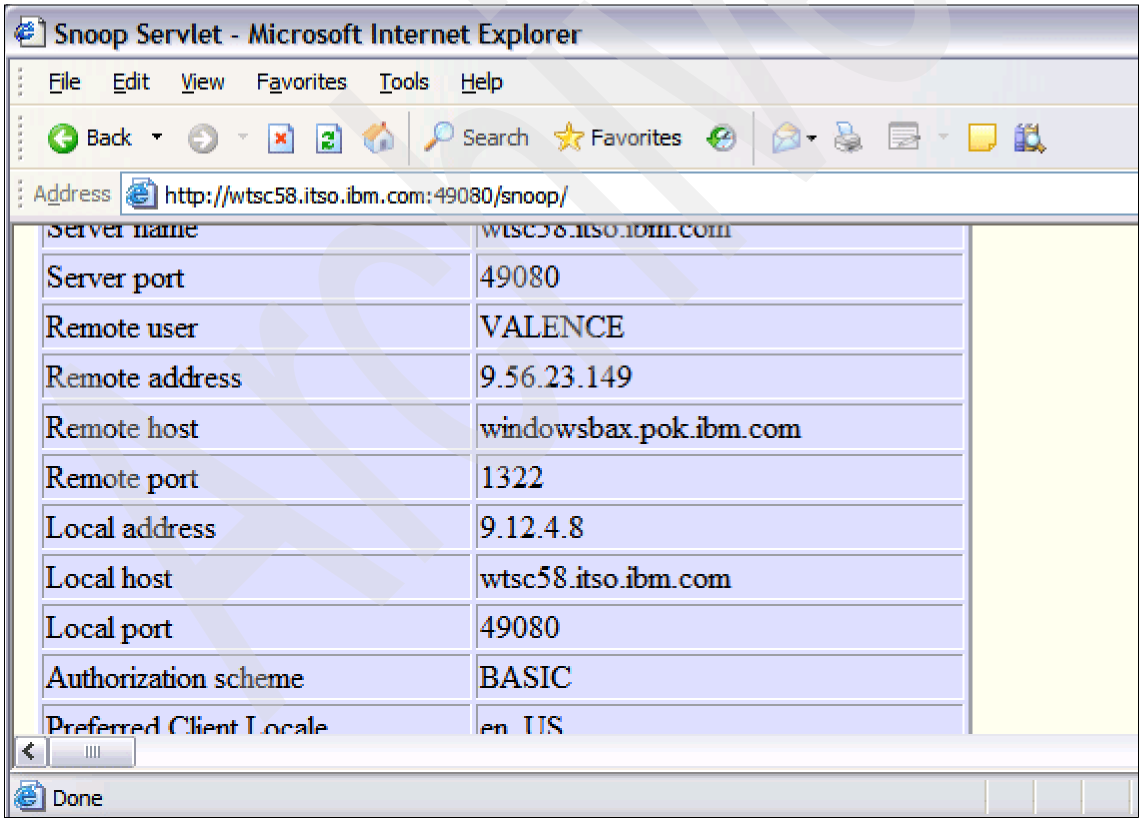
From a WebSphere Application Server for z/OS V7.0 perspective, using native authentication with z/OS LDAP is transparent. Consequently, the configuration is the same as with no native authentication. Refer to 12.3.2, “WebSphere and z/OS LDAP TDBM back end (DB2)" on page 447 to configure WebSphere Application Server.

WebSphere z/OS and LDAP native authentication validation

We validate the user registry used with the snoop servlet. With application security enabled, the snoop servlet requires basic authentication. Call the snoop servlet with a URL such as the following URL:

```
http://wtsc58.itso.ibm.com:49080/snoop/
```

Authenticate providing the user name and password defined earlier with the native authentication attribute and objectclass. The distinguished name (`cn=valence,ou=itsotdbm,o=itso`) or the common name only (`valence`) can be used. The RACF password corresponding to the ibm-nativeId attribute has to be provided. If the common name and the ibm-nativeId match, the credentials provided are RACF credentials. The snoop servlet then appears and shows the authenticated principal. Refer to Figure 12-11.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://wtsc58.itso.ibm.com:49080/snoop/
Done

Server name	wtsc58.itso.ibm.com
Server port	49080
Remote user	VALENCE
Remote address	9.56.23.149
Remote host	windowsbax.pok.ibm.com
Remote port	1322
Local address	9.12.4.8
Local host	wtsc58.itso.ibm.com
Local port	49080
Authorization scheme	BASIC
Preferred Client Locale	en_US

Figure 12-11 Snoop servlet showing authenticated z/OS LDAP TDBM native authentication user ID

12.4 Federated repositories

In this section, we explain federated repositories and how to configure them in our scenario.

12.4.1 Federated repositories

Inclusion of federated repositories in this WebSphere release provides a single model for managing organizational entities. You can configure a realm that consists of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in, file-based repository and in one or more external repositories.

Currently, most WebSphere applications have their own models and components for managing organizational entities, and they provide various levels of security. Most applications are dependent on specific types and brands of repositories, assume a specific schema for the data in those repositories, and are unable to use repositories with existing data. Federated repositories help these applications by providing them with a common model, secure access to various brands and types of repositories, and the ability to use repositories with existing data. The single model includes a set of organizational entity types and their properties, a repository-independent application programming interface (API), and a service provider programming interface (SPI) for plugging in repositories. XPath is the search language in the API and SPI.

The federated repository configuration uses multiple repositories simultaneously and recognizes the entries in the various repositories as entries representing distinct entities. By configuring an entry mapping repository, a federated repository configuration can use both LDAP and the database at the same time. The federated repository configuration hierarchy and constraints for identifiers provide the aggregated namespace for both of those repositories and prevent identifiers from colliding.

A federated repository configuration provides a property extension repository, which is a database regardless of the type of main profile repositories for a property-level join configuration. When an application uses the federated repository configuration to retrieve an entry for a person, the federated repository configuration transparently joins the properties of the person that is retrieved from either the LDAP or the client’s database with the properties of the person that is retrieved from the property extension repository into a single logical person entry.

When you configure a property extension repository, you can supply a valid data source, a direct connection configuration, or both. WebSphere first tries to connect by way of the data source. If the data source is not available, the system uses the direct access configuration.

Figure 12-12 presents the WebSphere federated repositories architecture overview. The federated repository feature is also called the *Virtual Member Manager (VMM)*.

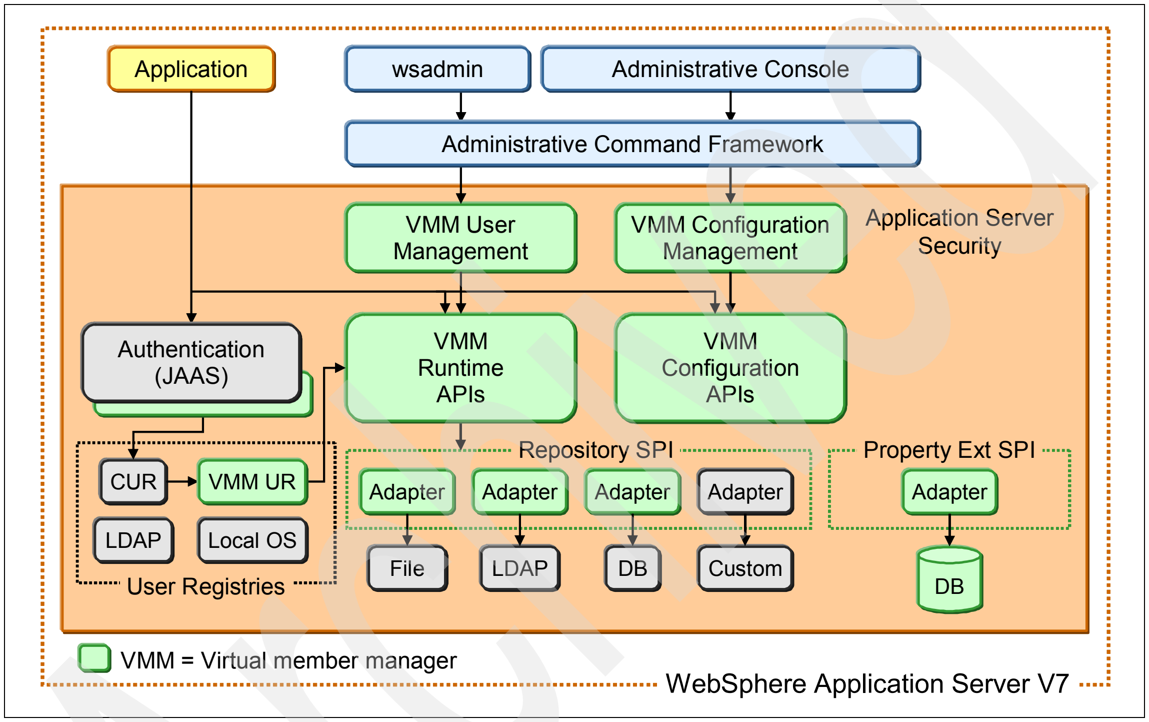


Figure 12-12 Federated repositories or VMM architecture overview

Federated repositories support the following user repositories in the cell's security realm:

- Built-in file-based repository
- Multiple federated LDAP servers
- Database that can be federated by the command-line interface (CLI)

  The federation creates a single namespace for identities. Database repositories are supported by using the command line only.

Federated repositories also provide user and group management features. The federated repository is accessed with read and write permissions from WebSphere Application Server. This user and group management is available through the administrative console, through command-line utilities, or using public APIs.

Table 12-1 shows a summary of federated repository features as compared to other user registry options.

Table 12-1 Federated repositories compared to other user registry options

	Federated repositories	**Other user registry options**
Supported registries	File-based LDAP DB (via `wsadmin`) RACF	Local operating system Stand-alone LDAP Stand-alone custom
Simultaneous multi-registry support	Yes	No
Registry read/write support	Read/write	Read-only

When you use the federated repositories functionality, all of the configured repositories that you specify as part of the federated repository configuration become active. *The user ID, and the distinguished name (DN) for an LDAP repository, must be unique in multiple user repositories that are configured under the same federated repository configuration.*

12.4.2 Our federated repositories scenario

Our federated repositories scenario relies on the LDAP tree and on the environment that we described in 12.2, “Our scenario and our environment” on page 437.

In this section, we focus on configuring WebSphere Application Server for z/OS for a federated repository composed of z/OS LDAP TDBM and IBM Tivoli Directory Server. Refer to Figure 12-13 on page 461.

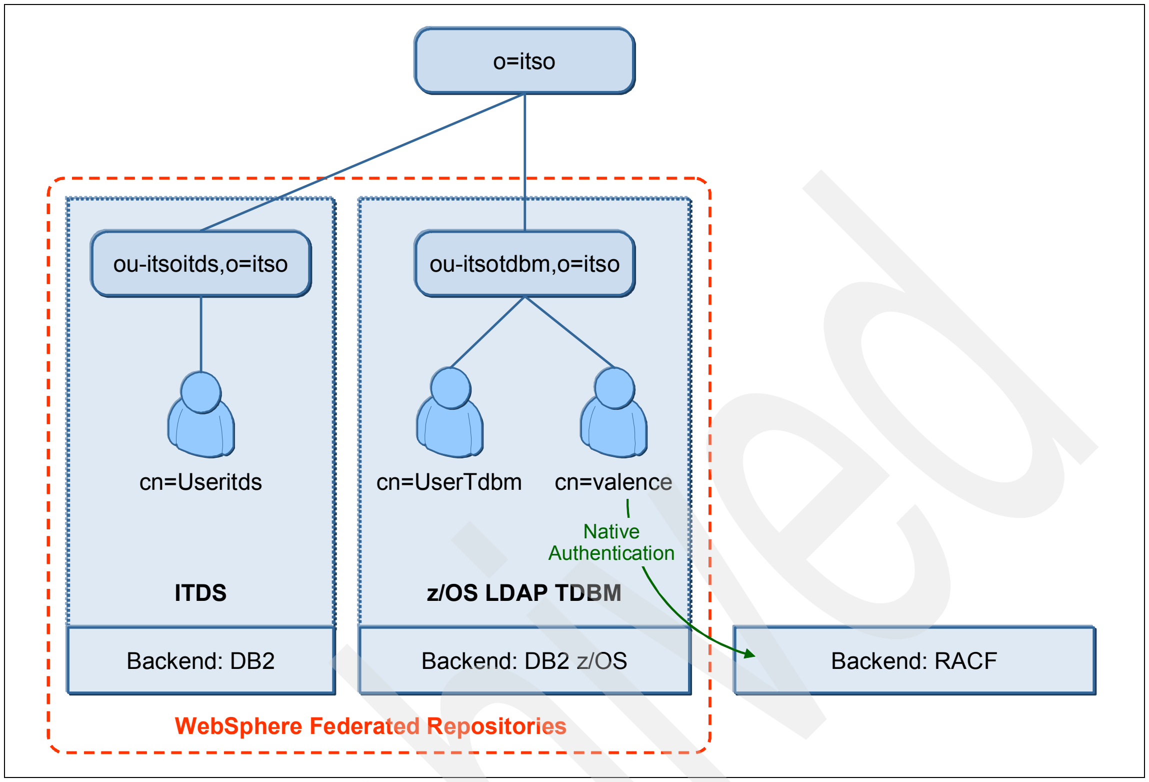


Figure 12-13 Federated repositories scenario

The `itsoitds` organizational unit is supported by IBM Tivoli Directory Server. The `itsotdbm` organizational unit is supported by z/OS LDAP TDBM. In this scenario, we federate the two subtrees to make them available to WebSphere as one LDAP tree whose root organization is `itso`. This federation is transparent to WebSphere Application Server for z/OS V7.0. Users and groups can be accessed in both subtrees simultaneously.

In the following section, we start the federation with z/OS LDAP TDBM. Then, we validate that it also works with native authentication, and finally we federate IBM Tivoli Directory Server. At the end of this federation configuration, all three users that are defined in our scenario can access the WebSphere application simultaneously.

12.4.3 Federated z/OS LDAP with TDBM back end (DB2)

In this section, we describe how to federate a z/OS LDAP with a TDBM back end.

Note: WebSphere Application Server for z/OS V7.0 does not currently support federated repositories with a z/OS LDAP *SDBM* back end. To access a z/OS LDAP SDBM back end, WebSphere can use a stand-alone LDAP registry configuration.

z/OS LDAP TDBM configuration

Configure z/OS LDAP with a TDBM back end, as described in 12.3.2, "WebSphere and z/OS LDAP TDBM back end (DB2)" on page 447.

WebSphere z/OS configuration for LDAP TDBM

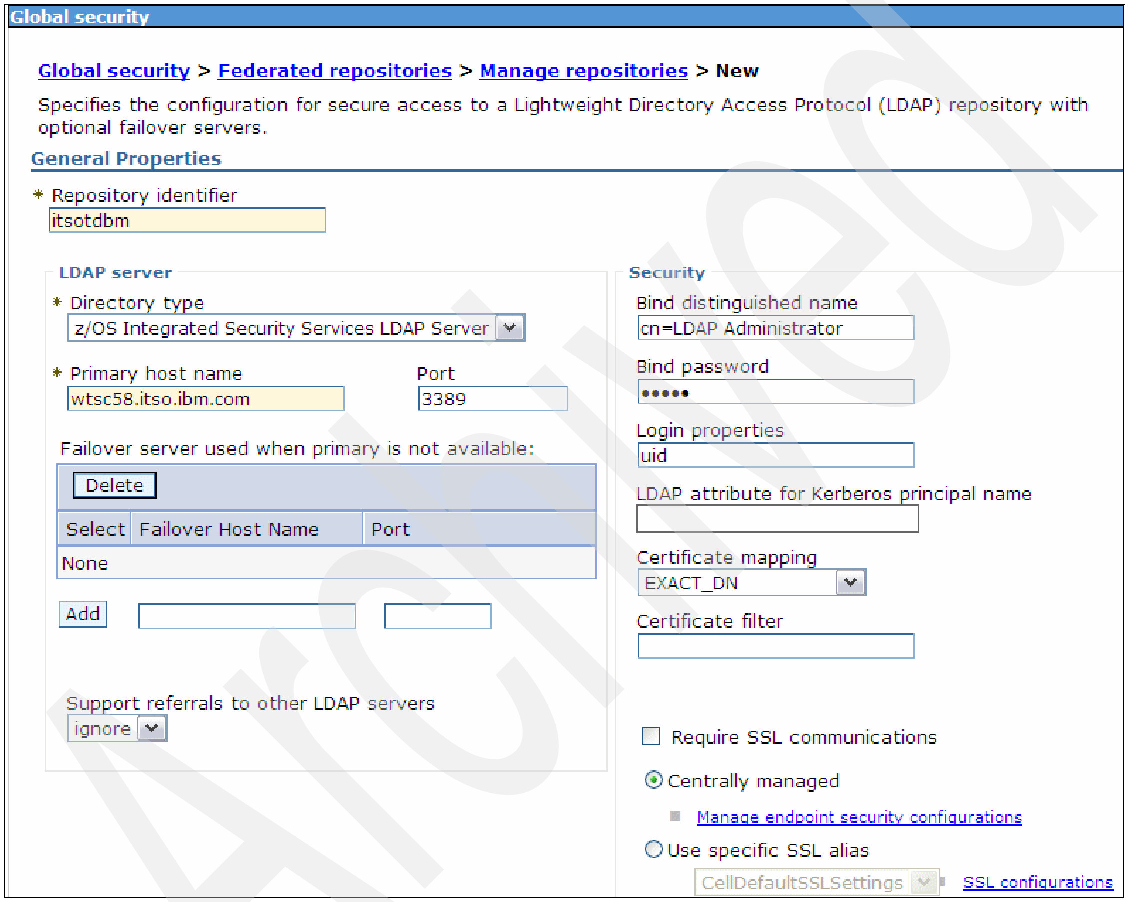
WebSphere Application Server for z/OS V7.0 supports accessing a z/OS LDAP with a TDBM back end (DB2) when configured as a federated repository LDAP registry. In this section, we explain how to configure WebSphere to access z/OS LDAP TDBM:

1. In the administrative console, select **Security** → **Global security**. Under User account repository, select **Federated repositories**, and then, click **Configure**. Under the Related Items section, select **Manage repositories**. The default InternalFileRepository appears in the list. Click **Add** to define a new repository.
2. Configure the new repository with the parameters for the z/OS LDAP TDBM back end (Figure 12-14 on page 463):
   - Repository identifier: Name of the repository in the WebSphere configuration. We choose `itsotdbm` in our configuration.
   - As a directory type, select **z/OS Integrated Security Services LDAP Server**.
   - Enter the primary host name and port for the z/OS LDAP server. These values are `wtsc58.itso.ibm.com` and `3389` in our environment.
   - It is possible to specify failover servers for high availability purposes.
   - Specify the bind distinguished name and password. This ID is an LDAP user ID that is allowed to scan and update the LDAP tree. We choose the administrator identity for our LDAP server in our environment, which is `cn=LDAP Administrator`.
   - For a TDBM back end with the schema loaded, leave `uid` in the Login properties field.

– Do not enter a value in the LDAP attribute for Kerberos principal name field.

– You can configure SSL to secure the connection to the LDAP server. We do not implement this feature in our environment.

Then, click **Apply**, and save to the master configuration. WebSphere validates that it can access the LDAP server.



Global security

Global security > Federated repositories > Manage repositories > New

Specifies the configuration for secure access to a Lightweight Directory Access Protocol (LDAP) repository with optional failover servers.

General Properties

* Repository identifier: itsotdbm

LDAP server

* Directory type: z/OS Integrated Security Services LDAP Server

* Primary host name: wtsc58.itso.ibm.com

Port: 3389

Failover server used when primary is not available:

Select	Failover Host Name	Port
None		

Support referrals to other LDAP servers: ignore

Security

Bind distinguished name: cn=LDAP Administrator

Bind password: •••••

Login properties: uid

LDAP attribute for Kerberos principal name:

Certificate mapping: EXACT_DN

Certificate filter:

☐ Require SSL communications

◉ Centrally managed

Manage endpoint security configurations

○ Use specific SSL alias

CellDefaultSSLSettings

SSL configurations

Figure 12-14 WebSphere z/OS LDAP TDBM as a federated repository

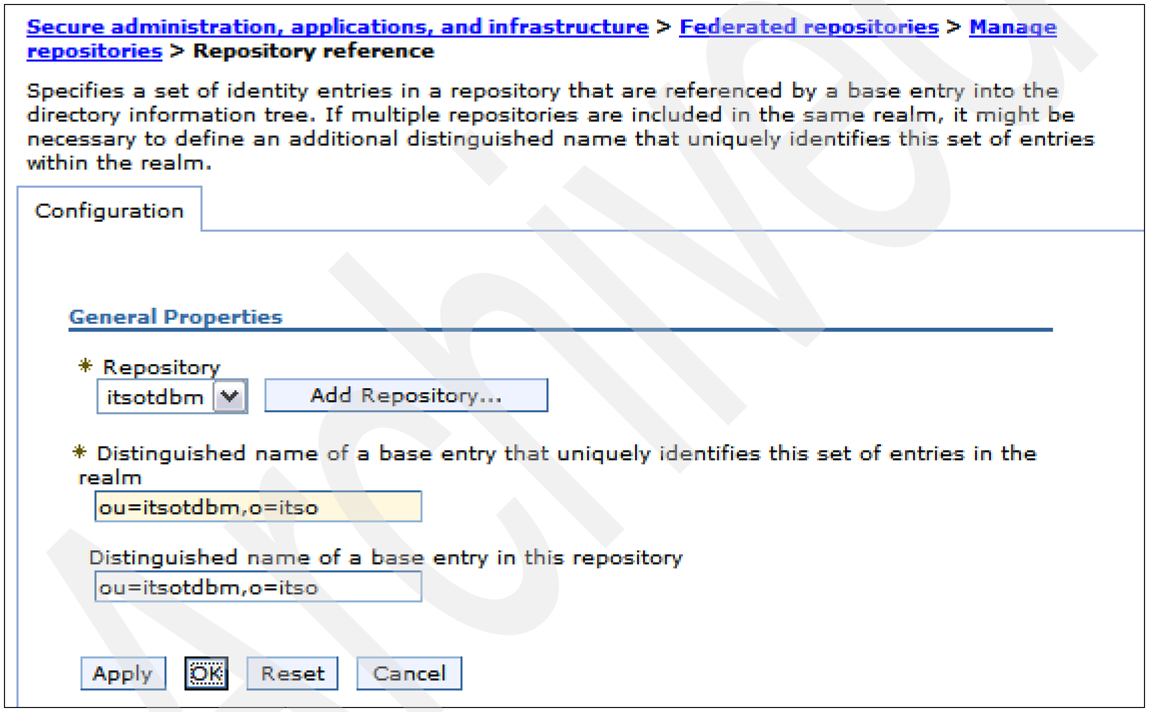
3. In the administrative console, select **Security** → **Global security**. Under User account repository, select **Federated repositories**, and then, click **Configure**. Under Repositories in the Realm, click **Add Base entry to Realm**:

   a. Select your new repository name. We use `itsotdbm` in our example.

b. Specify the distinguished name of a base entry that uniquely identifies this set of entries in the realm. If multiple repositories are included in the realm, you must define a separate distinguished name that uniquely identifies this set of entries within the realm. We choose `ou=itsotdbm,o=itso` in our configuration.

c. Specify the distinguished name of the base entry within the repository. The entry and its descendents are mapped to the subtree that is identified by the unique base name entry field. If this field is left blank, the subtree defaults to the root of the LDAP repository. We set up `ou=itsotdbm,o=itso` for our configuration.

Then, click **OK**, and save to the master configuration (Figure 12-15).



Secure administration, applications, and infrastructure > Federated repositories > Manage repositories > **Repository reference**

Specifies a set of identity entries in a repository that are referenced by a base entry into the directory information tree. If multiple repositories are included in the same realm, it might be necessary to define an additional distinguished name that uniquely identifies this set of entries within the realm.

Configuration

General Properties

* Repository: itsotdbm

* Distinguished name of a base entry that uniquely identifies this set of entries in the realm: ou=itsotdbm,o=itso

Distinguished name of a base entry in this repository: ou=itsotdbm,o=itso

Figure 12-15 WebSphere z/OS LDAP TDBM

4. In the administrative console, select **Security** → **Global security**. Under User account repository, select **Federated repositories**, and then, click **Configure**:

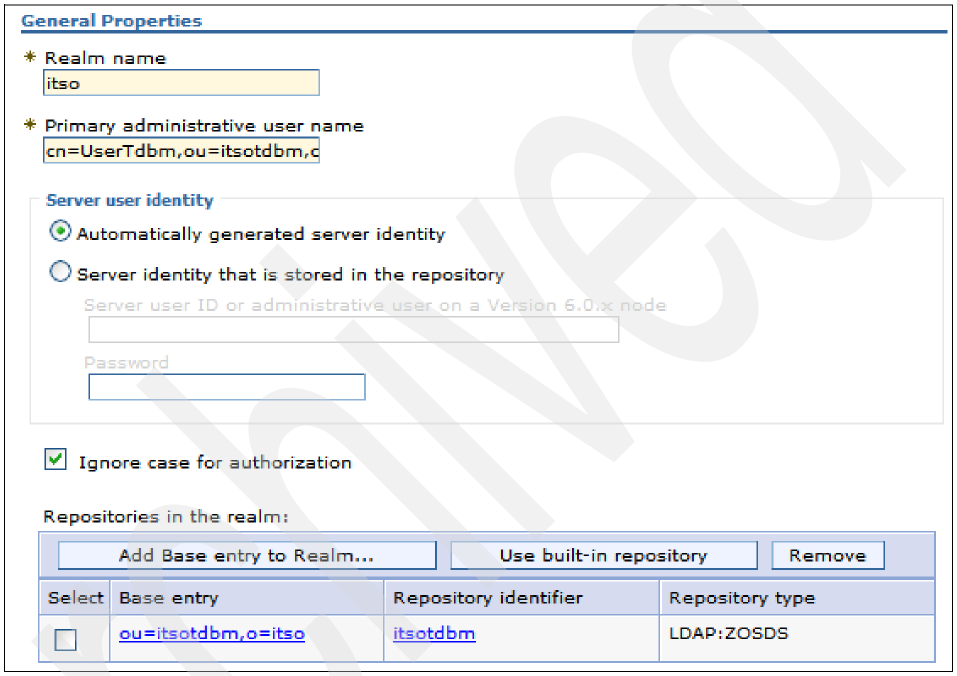
   a. Enter a realm name of your choice. We choose `itso` in our example.

   b. Specify the name of the user who will have WebSphere administrative privileges. This distinguished name has to be an existing identity in the z/OS LDAP TDBM repository. We choose `cn=UserTdbm,ou=itsotdbm,o=itso` in our environment.

c. Select **Automatically generated server identity**.

d. Remove the existing file-based InternalFileRepository by selecting it and clicking **Remove**.

Then, click **Apply**, and save to the master configuration. WebSphere validates that it can find the administrative user identity (Figure 12-16).


General Properties

* Realm name
itso

* Primary administrative user name
cn=UserTdbm,ou=itsotdbm,c

Server user identity
[● Automatically generated server identity]
○ Server identity that is stored in the repository
Server user ID or administrative user on a Version 6.0.x node
Password

[✓] Ignore case for authorization

Repositories in the realm:

Add Base entry to Realm... | Use built-in repository | Remove

Select	Base entry	Repository identifier	Repository type
☐	ou=itsotdbm,o=itso	itsotdbm	LDAP:ZOSDS

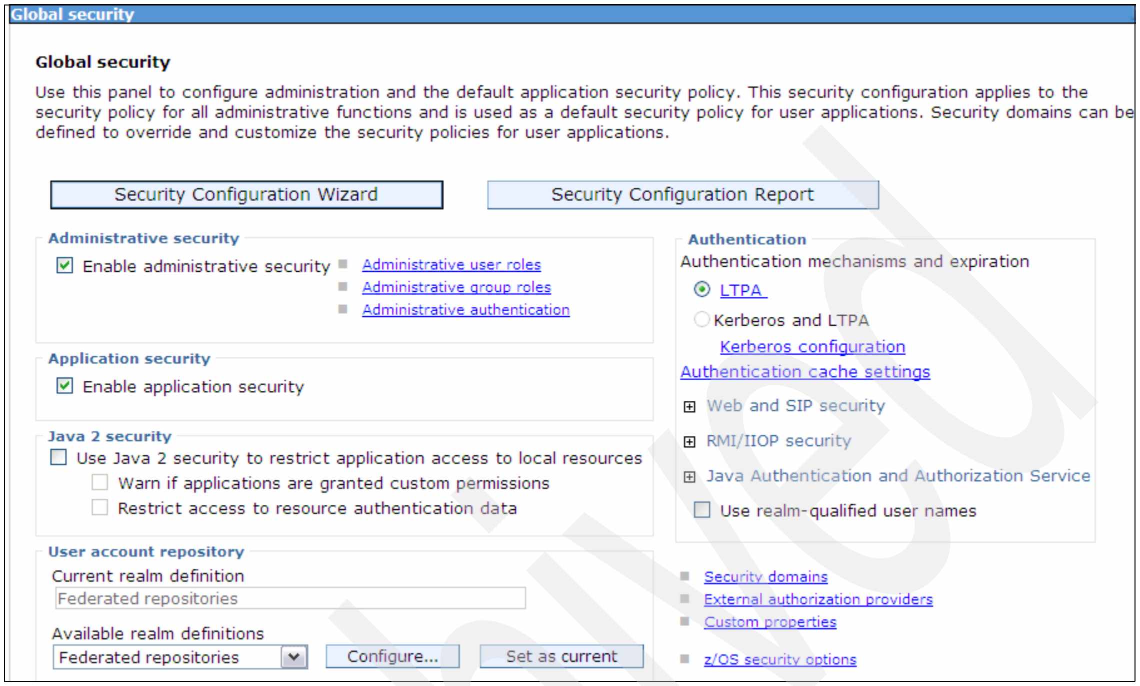
Figure 12-16 WebSphere federated repositories base entries

5. In the administrative console, select **Security** → **Global security**:

   a. Under User account repository, select **Federated repositories**, and then, click **Set as current**.

   b. Select **Administrative security** and clear the check mark from **Java2 security** if it is unnecessary.

c. Then, click **Apply**, and save to the master configuration (Figure 12-17).



Global security

Use this panel to configure administration and the default application security policy. This security configuration applies to the security policy for all administrative functions and is used as a default security policy for user applications. Security domains can be defined to override and customize the security policies for user applications.

Administrative security

- [✓] Enable administrative security
  - Administrative user roles
  - Administrative group roles
  - Administrative authentication

Application security

- [✓] Enable application security

Java 2 security

- [ ] Use Java 2 security to restrict application access to local resources
  - [ ] Warn if applications are granted custom permissions
  - [ ] Restrict access to resource authentication data

User account repository

Current realm definition: Federated repositories

Available realm definitions: Federated repositories

Authentication

Authentication mechanisms and expiration

- ● LTPA
- ○ Kerberos and LTPA
  - Kerberos configuration

Authentication cache settings

- [ ] Use realm-qualified user names

Global security
Security Configuration Wizard
Security Configuration Report
Configure...
Set as current
Web and SIP security
RMI/IIOP security
Java Authentication and Authorization Service
Security domains
External authorization providers
Custom properties
z/OS security options

Figure 12-17 WebSphere federated repositories main security panel

6. Restart WebSphere Application Server for z/OS V7.0.

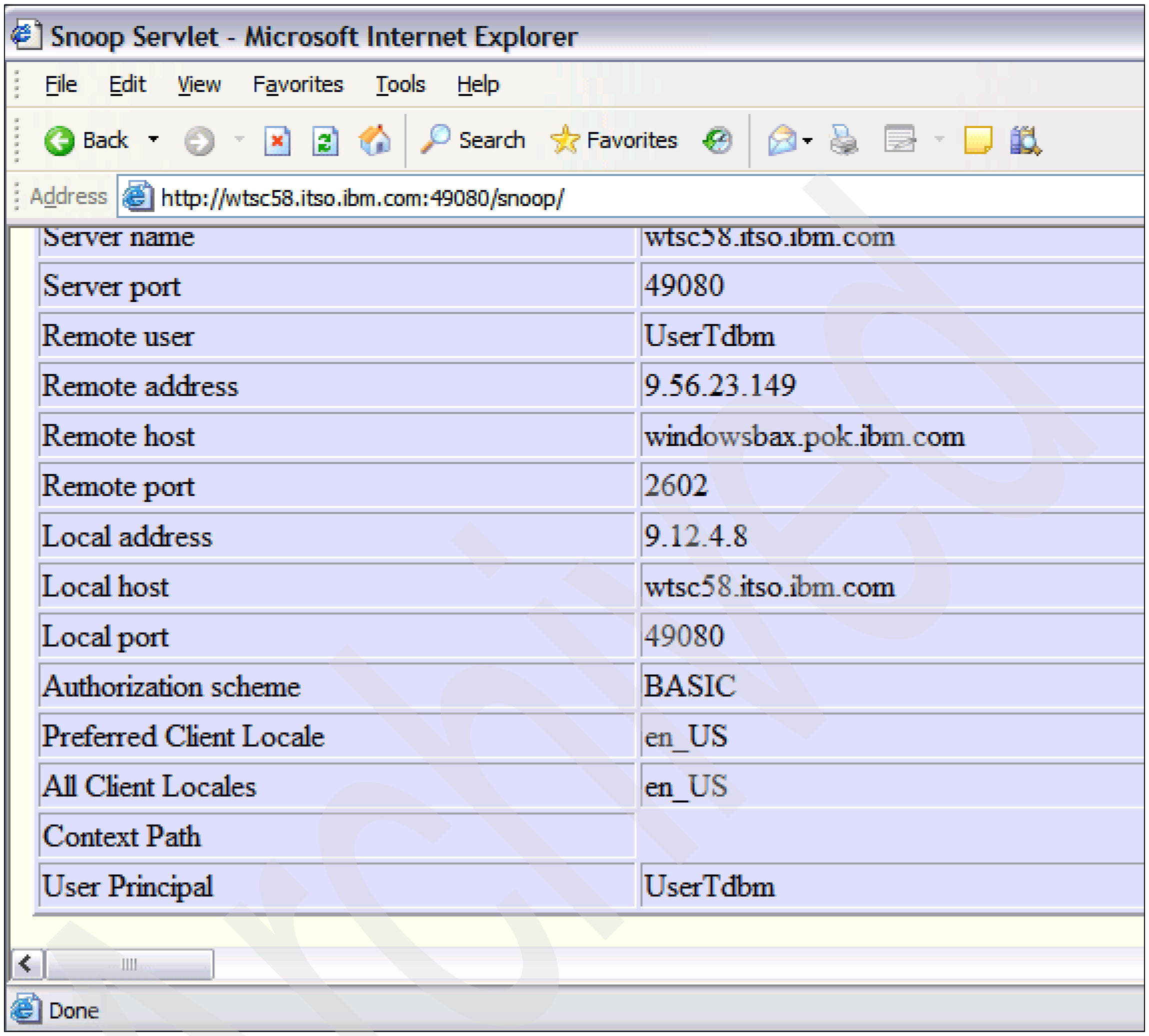
Federated z/OS LDAP TDBM validation

After restarting the cell, log in to the administrative console with the primary administrative user name that was defined earlier. You can use the full distinguished name (`cn=UserTdbm,ou=itsotdbm,o=itso`) or the user name only (`usertdbm`). Using our `ldif` file, the password is `usertdbm` also.

You can also validate the user registry that is used with the snoop servlet. With application security enabled, the snoop servlet requires basic authentication. Call the snoop servlet with a URL, such as the following URL:

```
http://wtsc58.itso.ibm.com:49080/snoop/
```

Authenticate providing the user name and password that were defined earlier. The snoop servlet then appears and shows the authenticated principal (Figure 12-18 on page 467).


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://wtsc58.itso.ibm.com:49080/snoop/
Done

Server name	wtsc58.itso.ibm.com
Server port	49080
Remote user	UserTdbm
Remote address	9.56.23.149
Remote host	windowsbax.pok.ibm.com
Remote port	2602
Local address	9.12.4.8
Local host	wtsc58.itso.ibm.com
Local port	49080
Authorization scheme	BASIC
Preferred Client Locale	en_US
All Client Locales	en_US
Context Path	
User Principal	UserTdbm

Figure 12-18 Snoop servlet showing authenticated z/OS LDAP TDBM user ID

12.4.4 Federated z/OS LDAP TDBM native authentication

In this section, we describe how to federate a z/OS LDAP TDBM with native authentication. We explain why you might choose to enable native authentication in 12.3.3, “WebSphere and z/OS LDAP TDBM native authentication” on page 454.

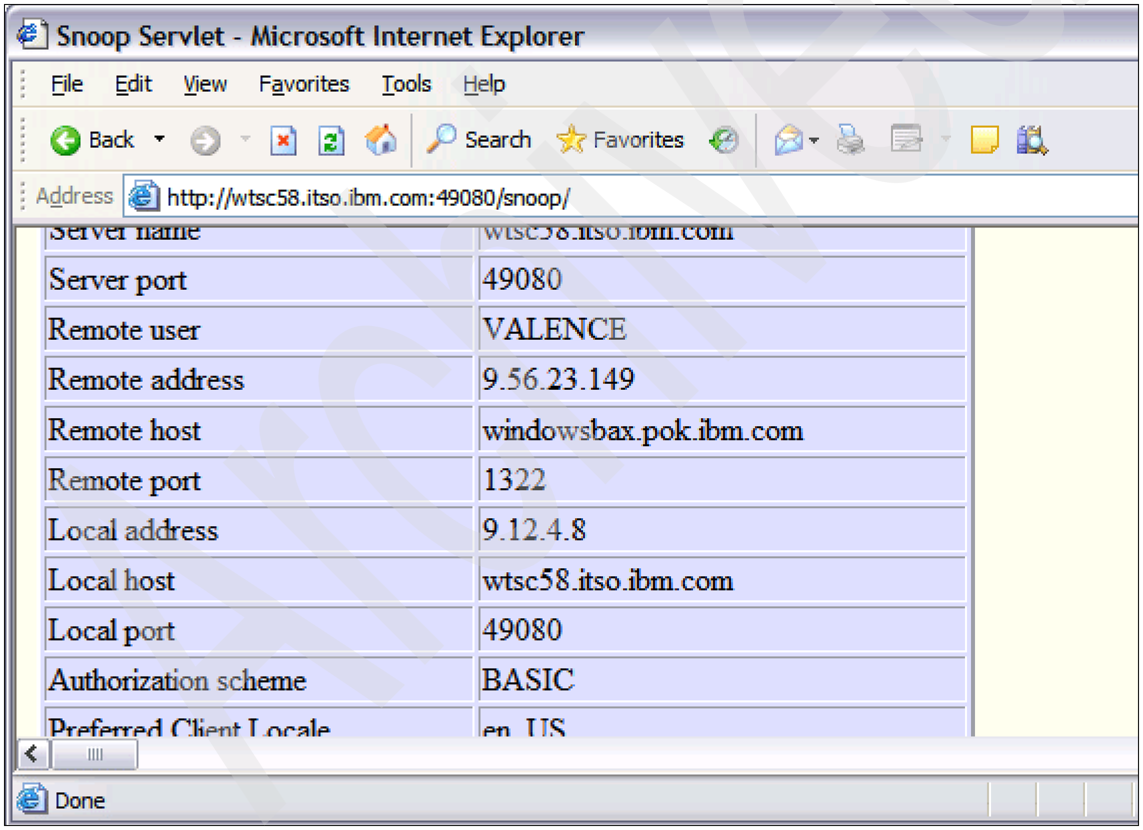
z/OS LDAP TDBM configuration

Configure z/OS LDAP with a TDBM back end, as described in 12.3.3, “WebSphere and z/OS LDAP TDBM native authentication” on page 454.

WebSphere z/OS configuration for LDAP TDBM

Native authentication does not imply configuration changes at the WebSphere level. Hence, configure WebSphere for z/OS for a z/OS LDAP TDBM back end, as described in 12.4.3, “Federated z/OS LDAP with TDBM back end (DB2)” on page 462.

You can now access your secured applications with the user IDs and password in RACF. For example, in our environment, the `valence` user can access the snoop servlet providing this user’s RACF password. Refer to Figure 12-19.

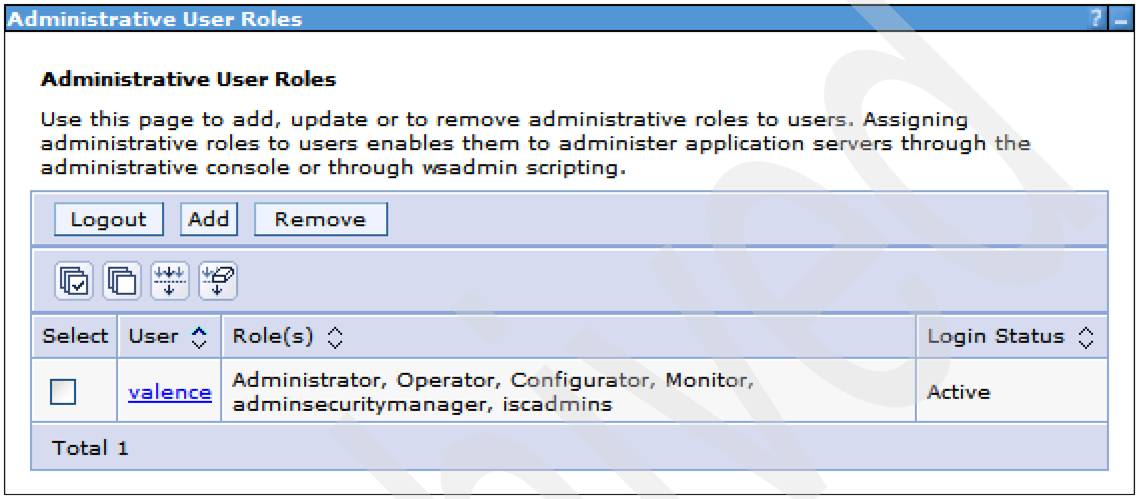

Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://wtsc58.itso.ibm.com:49080/snoop/
Done

Server name	wtsc58.itso.ibm.com
Server port	49080
Remote user	VALENCE
Remote address	9.56.23.149
Remote host	windowsbax.pok.ibm.com
Remote port	1322
Local address	9.12.4.8
Local host	wtsc58.itso.ibm.com
Local port	49080
Authorization scheme	BASIC
Preferred Client Locale	en_US

Figure 12-19 Snoop servlet showing authenticated z/OS LDAP TDBM native authorization user ID

In order to access the administrative console, users need to be added to the list of administrative roles. The user can be added in **Users and Groups → Administrative user roles.**

For example, access the administrative console with the existing administrative user ID (`usertdbm` in our example), and then, add any new user (`valence` in our example) with an administrative role. Refer to Figure 12-20.



Administrative User Roles

Use this page to add, update or to remove administrative roles to users. Assigning administrative roles to users enables them to administer application servers through the administrative console or through wsadmin scripting.

Select	User	Role(s)	Login Status
☐	valence	Administrator, Operator, Configurator, Monitor, adminsecuritymanager, iscadmins	Active

Total 1

Figure 12-20 WebSphere administrative user roles

12.4.5 Federated IBM Tivoli Directory Server

In this section, we describe how to federate a IBM Tivoli Directory Server.

IBM Tivoli Directory Server configuration

The Tivoli Information Center describes how to install IBM Tivoli Directory Server:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDS.doc/install04.htm`

In this section, we focus on the configuration activities.

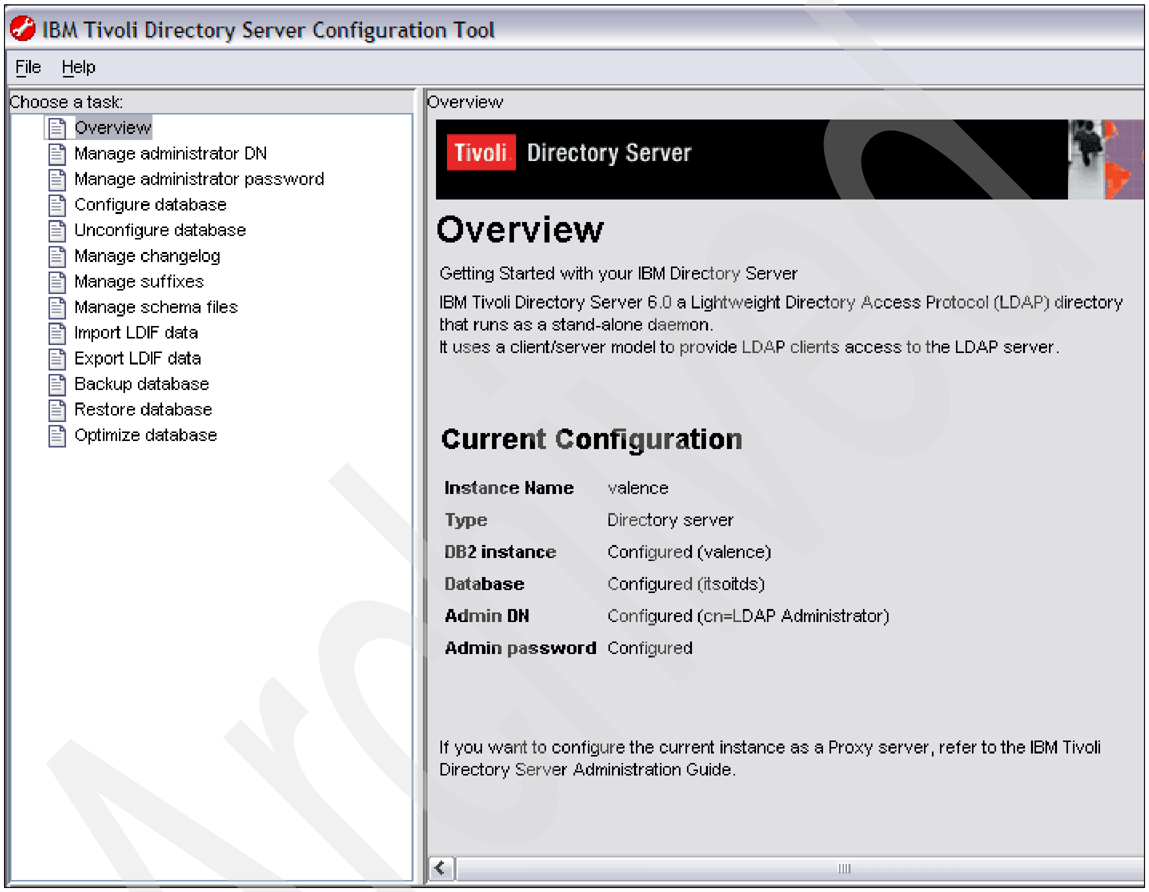
On the Windows platform, you can configure IBM Tivoli Directory Server by using the idsxcfg utility. This utility is available using the following command:

```
C:\Program Files\IBM\LDAP\V6.0\sbin\idsxcfg.cmd
```

We configure IBM Tivoli Directory Server to handle the following suffix:

```
o=itsoitds,o=itso
```

The LDAP administrator is `cn=LDAP Administrator`, and the password is `secret`. Refer to Figure 12-21.



Overview

Getting Started with your IBM Directory Server

IBM Tivoli Directory Server 6.0 a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon.
It uses a client/server model to provide LDAP clients access to the LDAP server.

Current Configuration

Instance Name	valence
Type	Directory server
DB2 instance	Configured (valence)
Database	Configured (itsoitds)
Admin DN	Configured (cn=LDAP Administrator)
Admin password	Configured

If you want to configure the current instance as a Proxy server, refer to the IBM Tivoli Directory Server Administration Guide.

Figure 12-21 IBM Tivoli Directory Server idsxcfg current configuration display

We now add entries to IBM Tivoli Directory Server, because it is empty. We add a suffix entry and a person entry. For this purpose, create a new `schema.suffix.ldif` that contains the following information:

```
dn: ou=itsoitds,o=itso
ou: itsoitds
objectclass: organizationalUnit
objectclass: top
```

```
dn: cn=UserItds, ou=itsoitds,o=itso
uid: UserItds
description: Test user for ITDS
objectclass: inetOrgPerson
objectclass: ePerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
sn: 2007
cn: UserItds
```

Customize these entries to match your suffix and the user name that you want for a first user.

Use a command similar to the following command to add the entries to the LDAP tree:

```
C:\Program Files\IBM\LDAP\V6.0\bin>ldapadd -D "cn=LDAP Administrator"
-w secret -i schema.suffix.ldif
```

To set up a password for the new user, create a file called `user.password.ldif`, which contains the following information:

```
dn: cn=UserItds,ou=itsoitds,o=itso
changetype:modify
replace:userpassword
userpassword: useritds
```

Use a command similar to the following command to modify the user entry in the LDAP tree:

```
C:\Program Files\IBM\LDAP\V6.0\bin>ldapmodify -D "cn=LDAP
Administrator" -w secret -i user.password.ldif
```

WebSphere z/OS configuration for IBM Tivoli Directory Server

WebSphere Application Server for z/OS V7.0 supports accessing IBM Tivoli Directory Server when configured as a federated repository LDAP registry. In this section, we explain how to configure WebSphere to access IBM Tivoli Directory Server:

1. In the administrative console, select **Security** → **Global security**. Under User account repository, select **Federated repositories**, and then, click **Configure**. Under the Related Items section, select **Manage repositories**. The default InternalFileRepository appears in the list. Click **Add** to define a new repository.

2. Configure the new repository with the parameters for IBM Tivoli Directory Server (Figure 12-22 on page 473):

   – Repository identifier: Name of the repository in the WebSphere configuration. We choose `itsoitds` in our configuration.

   – As a directory type, select **IBM Tivoli Directory Server**.

   – Enter the primary host name and port for the z/OS LDAP server. We use `itds.itso.ibm.com` and `389` in our environment.

   – You can specify failover servers for high availability purposes.

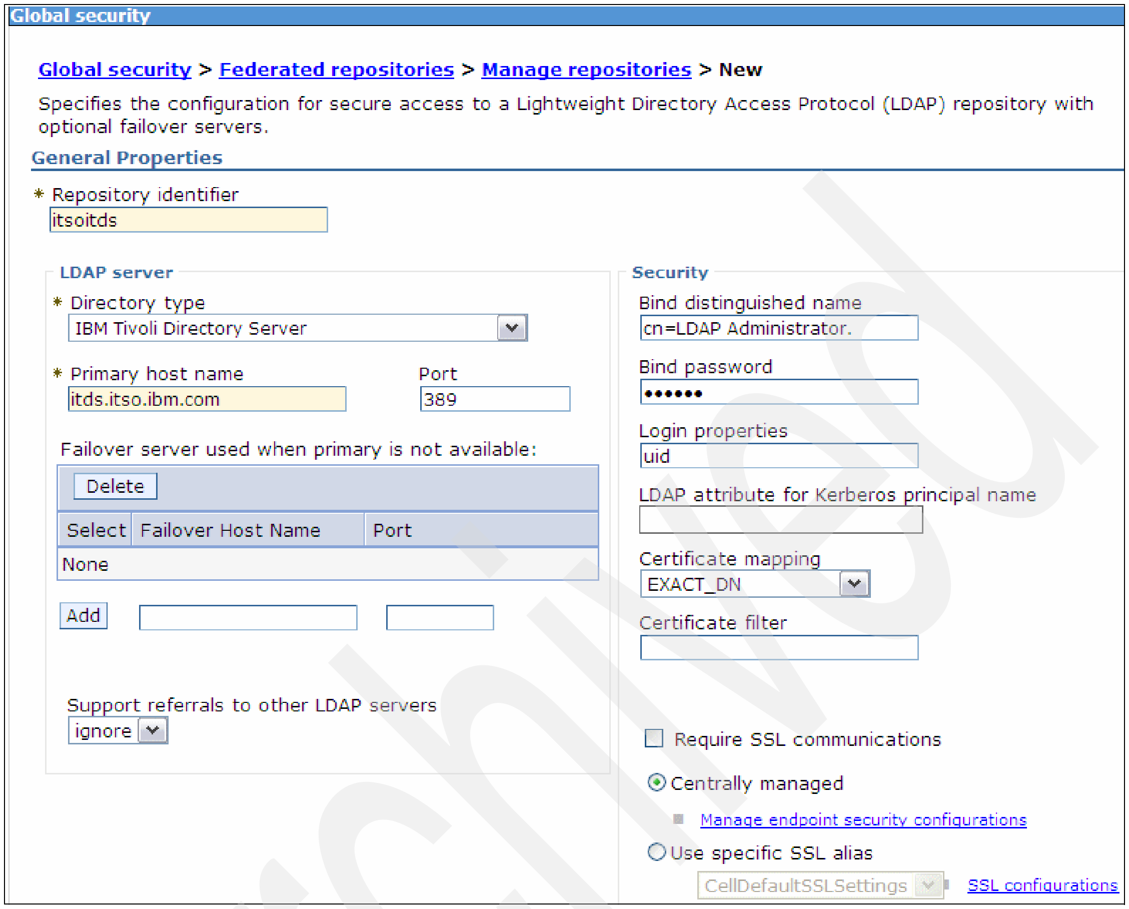
   – Specify the bind distinguished name and password, which needs to be an LDAP user ID that is allowed to scan and update the LDAP tree. We choose the administrator identity for our LDAP server in our environment, which is `cn=LDAP Administrator`.

   – Leave `uid` in the Login properties field.

   – The LDAP attribute for Kerberos principal name field is protected, and it is not possible to assign a value.

   – You can configure SSL to secure the connection to the LDAP server. We do not implement this feature in our environment.

   Then, click **Apply**, and save to the master configuration. WebSphere validates that it can access the LDAP server.


Global security

Global security > Federated repositories > Manage repositories > New

Specifies the configuration for secure access to a Lightweight Directory Access Protocol (LDAP) repository with optional failover servers.

General Properties

* Repository identifier: itsoitds

LDAP server

* Directory type: IBM Tivoli Directory Server

* Primary host name: itds.itso.ibm.com

Port: 389

Failover server used when primary is not available:

Select	Failover Host Name	Port
None		

Support referrals to other LDAP servers: ignore

Security

Bind distinguished name: cn=LDAP Administrator.

Bind password: ●●●●●●

Login properties: uid

LDAP attribute for Kerberos principal name:

Certificate mapping: EXACT_DN

Certificate filter:

☐ Require SSL communications

● Centrally managed

Manage endpoint security configurations

○ Use specific SSL alias

CellDefaultSSLSettings

SSL configurations

Figure 12-22 WebSphere configuration for IBM Tivoli Directory Server as a federated repository

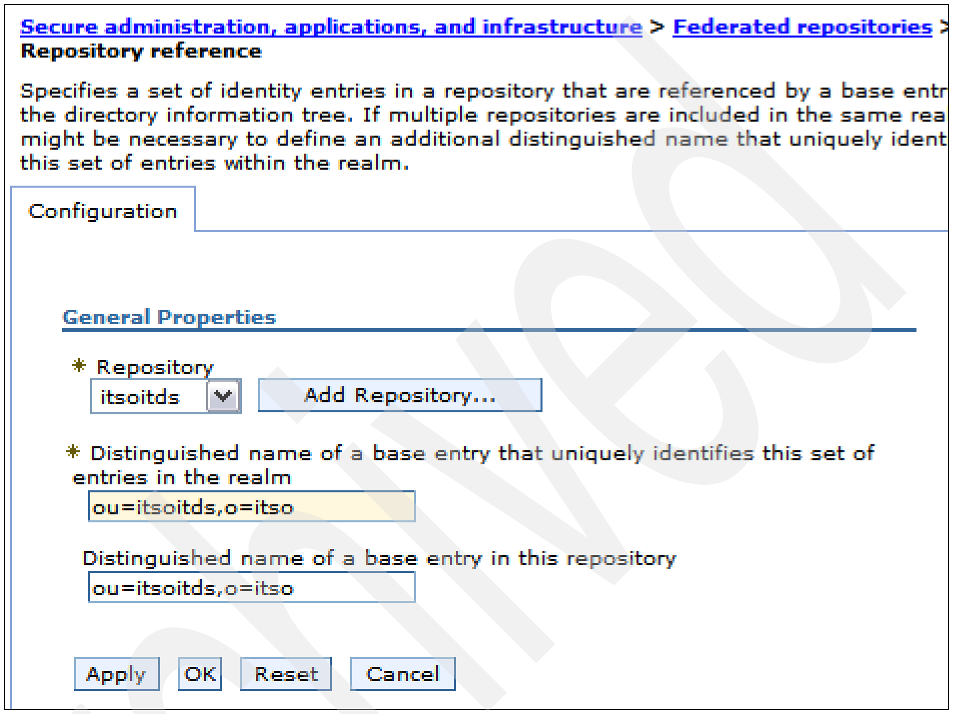
3. In the administrative console, select **Security** → **Global security**. Under User account repository, select **Federated repositories**, and then, click **Configure**. Under Repositories in the Realm, click **Add Base entry to Realm**:

   – Select your new repository name. We use `itsoitds` in our example.

   – Specify the distinguished name of a base entry that uniquely identifies this set of entries in the realm. If multiple repositories are included in the realm, it is necessary to define a separate distinguished name that uniquely identifies this set of entries within the realm. We choose `ou=itsoitds,o=itso` in our configuration.

   – Specify the distinguished name of the base entry within the repository. The entry and its descendents are mapped to the subtree that is identified by the unique base name entry field. If this field is left blank, the subtree defaults to the root of the LDAP repository. We set up `ou=itsoitds,o=itso` for our configuration.

Then, click **OK**, and save to the master configuration (Figure 12-23).



Secure administration, applications, and infrastructure > Federated repositories >
Repository reference

Specifies a set of identity entries in a repository that are referenced by a base entr
the directory information tree. If multiple repositories are included in the same rea
might be necessary to define an additional distinguished name that uniquely ident
this set of entries within the realm.

Configuration

General Properties

* Repository
itsoitds

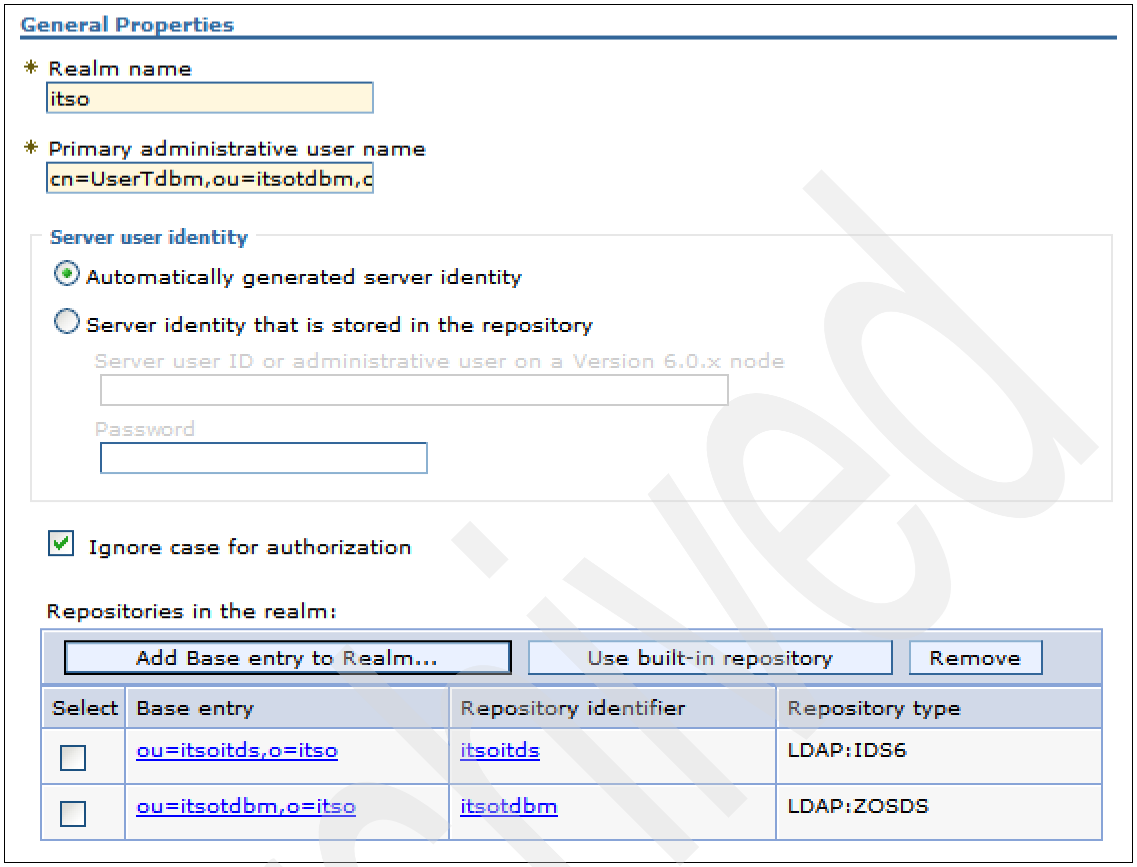
* Distinguished name of a base entry that uniquely identifies this set of entries in the realm
ou=itsoitds,o=itso

Distinguished name of a base entry in this repository
ou=itsoitds,o=itso

Figure 12-23 WebSphere repository reference for IBM Tivoli Directory Server

4. In the administrative console, select **Security** → **Global security**. Under User account repository, select **Federated repositories**, and then, click **Configure**.

   In the example configuration, we federate z/OS LDAP with IBM Tivoli Directory Server. Because we have already configured this panel for z/OS LDAP, we do not need to perform any further configuration. Therefore, the primary administrative user name stays a user name from our first registry (`cn=UserTdbm,ou=itsotdbm,o=itso`). The realm name is the name for all federated registries. Refer to Figure 12-24 on page 475.



General Properties

* Realm name

itso

* Primary administrative user name

cn=UserTdbm,ou=itsotdbm,c

Server user identity

(●) Automatically generated server identity

( ) Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

[✓] Ignore case for authorization

Repositories in the realm:

Select	Base entry	Repository identifier	Repository type
☐	ou=itsoitds,o=itso	itsoitds	LDAP:IDS6
☐	ou=itsotdbm,o=itso	itsotdbm	LDAP:ZOSDS

Figure 12-24 WebSphere federated repositories base entries

5. In the administrative console, click **Security** → **Global security**.

   If you have not already done so, under User account repository, select **Federated repositories**, and then, click **Set as current**. Select **Administrative security**, and clear the check mark from Java2 security if it is unnecessary. Then, click **Apply**, and save to the master configuration.

6. Restart WebSphere Application Server for z/OS V7.0.

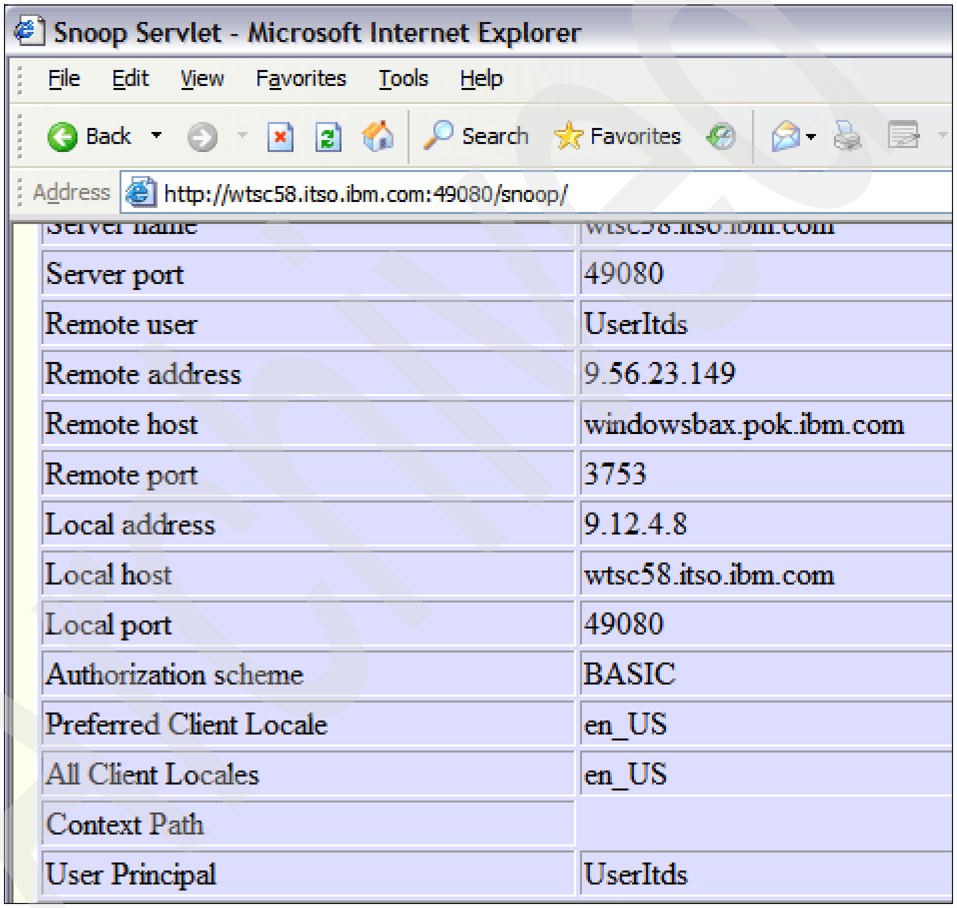
Federated IBM Tivoli Directory Server validation

To log in to the administrative console, use the primary administrative user name that was defined earlier. You can use the full distinguished name (`cn=UserTdbm,ou=itsotdbm,o=itso`) or the user name only (`usertdbm`). Using our `ldif` file, the password is `usertdbm` also.

You can validate the federated user registry with the snoop servlet, which is bundled in WebSphere for z/OS. With the application security enabled, the snoop servlet requires basic authentication. Call the snoop servlet with a URL, such as the following URL:

```
http://wtsc58.itso.ibm.com:49080/snoop/
```

Authenticate, providing the user name and password that were defined earlier for the last federated repository (`UserItds` in our example). The snoop servlet then appears and shows the authenticated principal. Refer to Figure 12-25.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wtsc58.itso.ibm.com:49080/snoop/

Server port	49080
Remote user	UserItds
Remote address	9.56.23.149
Remote host	windowsbax.pok.ibm.com
Remote port	3753
Local address	9.12.4.8
Local host	wtsc58.itso.ibm.com
Local port	49080
Authorization scheme	BASIC
Preferred Client Locale	en_US
All Client Locales	en_US
Context Path	
User Principal	UserItds

Figure 12-25 Snoop servlet showing authenticated IBM Tivoli Directory Server user ID

Because user registries are now federated, the user from z/OS LDAP TDBM (`UserTdbm`) can also be used with the same configuration at the same time.

All of these steps validate the federated repositories set up with a z/OS LDAP TDBM and IBM Tivoli Directory Server. Native authentication can also be used.

13

Implementing Secure Sockets Layer in WebSphere for z/OS

This chapter explains the concepts of certificates and keyrings. In the past, many administrators have had problems establishing Secure Sockets Layer (SSL) connections, for instance, when invoking the **wsadmin** shell or by federating a new node. Several publications even recommended that you turn off security for federating a node. In order to work with a secured WebSphere environment on z/OS, it is essential to understand the concepts of certificates and keyrings.

The new writable keyring support is also discussed in this chapter. This feature enhances the capabilities of the administrative console to consistently manage the keyrings and certificates that are stored in System Authorization Facility (SAF). This major enhancement increases the efficiency of SSL management within WebSphere Application Server for z/OS while SAF still has the control over all certificates and keyrings.

This chapter contains the following sections:

- "RACF certificate management" on page 514
- "Hardware cryptography and Java cryptography providers" on page 519
- "SSL troubleshooting and traces" on page 533

13.1 Keyrings and certificates in RACF

Getting connected to a secured cell with administrative security enabled can sometimes seem difficult, especially when federating a node into a existing cell. In many client cases, the SSL connection cannot be established because of missing keyring definitions and certificates.

Certain publications have recommended that you turn off security when federating a node into a existing cell. But this workaround for the federation cannot be used in large production environments with short time slots for maintenance. For normal configuration changes with **`wsadmin`** scripting, turning off the security is not a option either.

In order to work with a secured WebSphere environment on z/OS, it is essential to understand the concepts of certificates and keyrings.

13.1.1 Certificates

A *personal certificate* represents the identity of a server. The certificate has been certified by a Certification Authority (CA), ensuring that the server presenting the certificate is who it pretends to be. The personal certificate consists of a public/private key pair that is usually signed by a trusted CA. The entire concept of asymmetric cryptography is based on the fact that a message, which is signed by a public key, can only be decrypted by the corresponding private key and reverse. In practice, the private key will kept secret whereas the public key is made publicly available.

The major task of a trusted CA is to map an identity, such as a host name, to a specific public/private key pair in order to build trust. In addition, the CA itself consists of a self-signed public/private key pair, where the private key is the most secret part of this concept. A personal certificate is signed with the private key of the CA. Consequently, the trust of a personal certificate can be verified by using the public certificate of the CA, which is a *signer certificate*.

Essential to this concept is that you always have to trust the CA. A lot of clients have implemented their own CAs.

13.1.2 Keyrings

A *keyring* is a named collection of certificates associated with a specific user. A certificate is identified by its label, as well as the keyring to which it is connected. Consequently, a keyring is only valid in conjunction with a RACF user ID. You can create multiple keyrings with the same name, but assigned to separate user IDs. Refer to Figure 13-1.

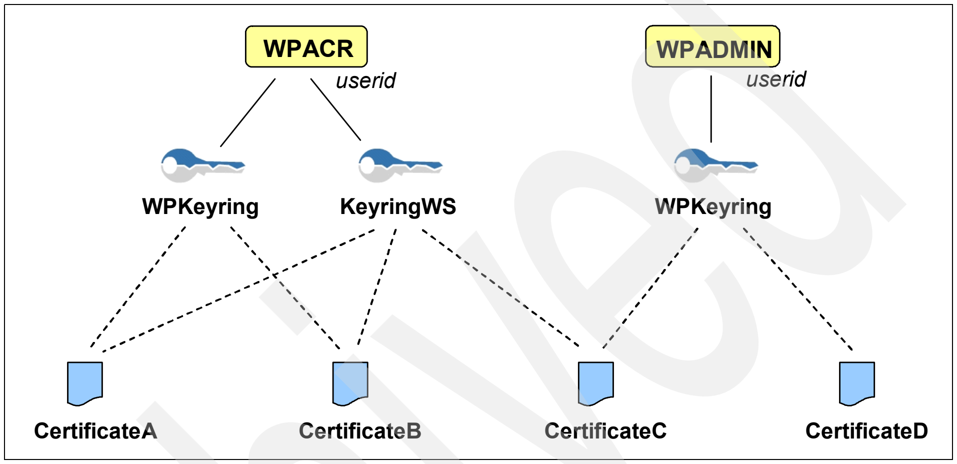


Figure 13-1 Keyring concept

For instance, a keyring named `WPKeyring` in conjunction with the started task control (STC) user ID `WPACR` differs from the keyring `WPKeyring` in conjunction with user ID `WPADMIN` as shown in Figure 13-1. Certificates can be connected to both keyrings independently.

13.1.3 Building a trust chain with WebSphere for z/OS

In order to explain how a trust chain works with regard to WebSphere Application Server on z/OS, this section uses an example that is based on the scenario that is seen in Figure 13-2 on page 483. The concepts in this scenario can be applied to any kind of SSL connection between client and server.

In the example, there are two situations that use SSL:

- A client initiates an SSL connection by invoking an application on WebSphere for z/OS. In this case, the application server on z/OS has the role of a server.
- The application needs to make outbound Web service requests to a second WebSphere environment on a distributed platform secured by SSL (we refer

to this environment as `dWAS`). In this case, the application server on z/OS has the role of the client and `dWAS` has the role of server.

In both cases, the initiator of the SSL connection must be able to trust the server.

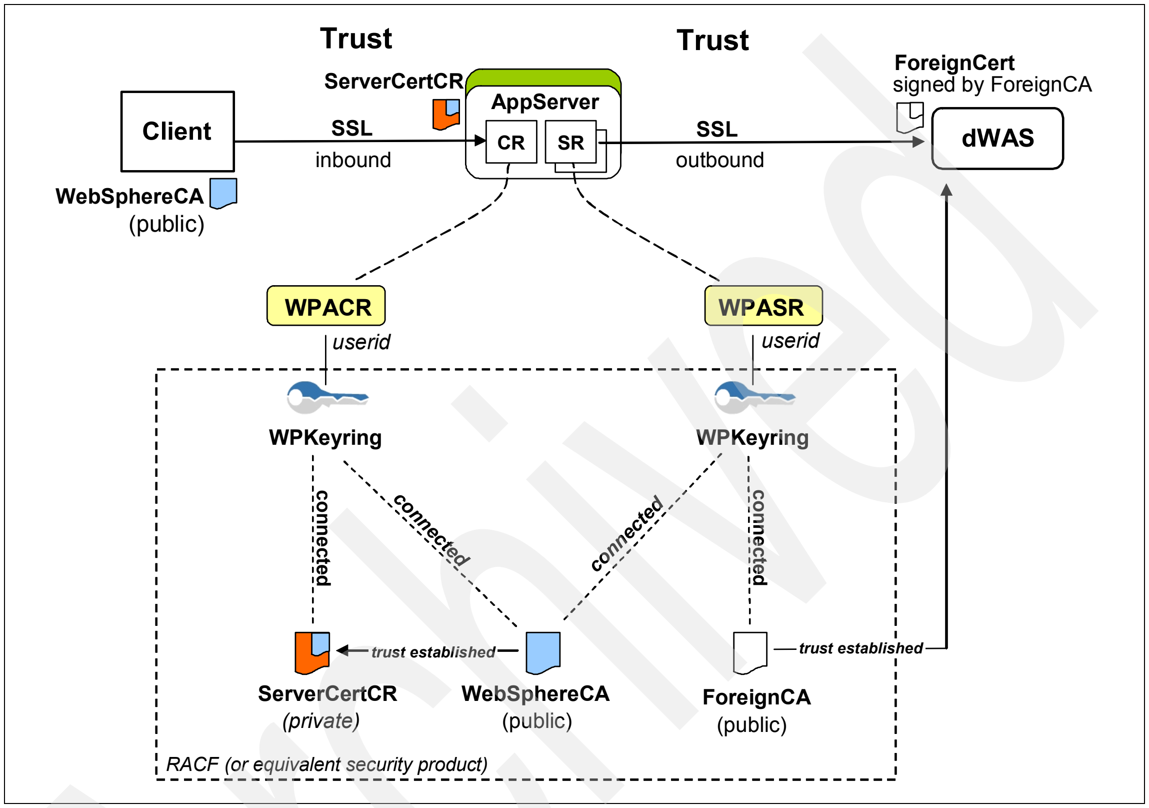


Figure 13-2 Building a trust chain with WebSphere z/OS

All incoming HTTP and SOAP requests (inbound) to the application server on z/OS are handled by the control region (CR) of the application server. It is the responsibility of the CR to establish the SSL connection with the client.

In contrast, the servant region (SR) of the application server is responsible for any outbound connections to other systems, including the SSL connection to `dWAS`.

Typically, CRs and SRs have separate started task control (STC) user IDs, and consequently, they have separate keyrings. It is common to use the same keyring name across the node. Because each keyring is assigned to a separate user ID, they are completely independent from each other.

In this example, the signer certificate of the WebSphereCA is connected to both keyrings. Additionally, a personal server certificate labeled `ServerCertCR` is connected to keyring `WPkeyring` of the CR. This certificate is signed by the `WebSphereCA`. An additional signer certificate of `ForeignCA` is connected to the keyring of the SR.

To establish the SSL connection between the client and the z/OS application server, the personal certificate `ServerCertCR`, which is connected to the keyring of the CR, will be used. In order to trust the server certificate, the client has to verify the signature using the `WebSphereCA` signer certificate. It is essential for that trust relationship that the client has imported the `WebSphereCA` signers certificate into its browser.

In the second case, the application server on z/OS is the initiator of the SSL connection and is the client. For this SSL connection, `dWAS` provides a server certificate labeled `ForeignCert`, which is signed by `ForeignCA`. In that trust relationship, it is essential that the application server trusts the server certificate of `dWAS`. Therefore, it is necessary to import a signer certificate of `ForeignCA` into RACF and connect it to the keyring of the SR.

13.1.4 Establishing SSL connections in an secured environment

The scenario in Figure 13-3 on page 485 illustrates how SSL connections between nodes are established. This scenario will be of interest to clients who have had problems establishing SSL connections when federating a new node into a secured environment or using the **`wsadmin`** shell of a federated node.

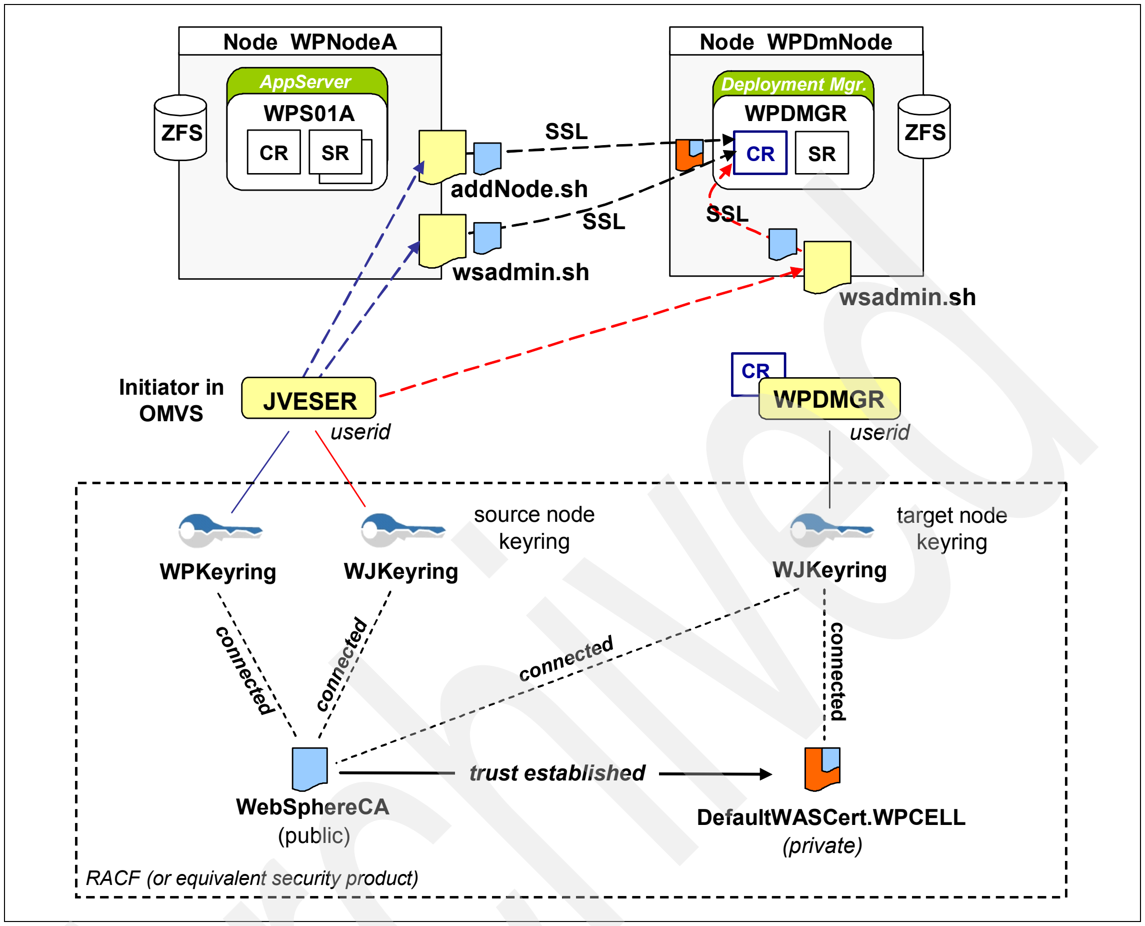


Figure 13-3 Keyrings and certificates using wsadmin

In order to establish an SSL connection, it is important that the connection initiator can trust the server.

In this example, the server is the deployment manager and the initiator is the OMVS user ID `JVESER`. The initiator is executing the **`wsadmin`** or addNode shell scripts from another WebSphere node.

The user ID used to authenticate to the deployment manager and the user ID invoking the shell script can differ. With regard to the SSL connection, only the user ID executing the shell script is relevant. Example 13-1 on page 486 shows a **`wsadmin`** shell script invocation from an application server node to the deployment manager. Only the bold marked user ID is of importance for the SSL connection, not the user ID and password passed to the **`wsadmin`** shell script.

Example 13-1 Sample invocation of wsadmin shell script

```
JVESER @ SC04:/wasconfig/wpcell/wpnodea/AppServer/bin>wsadmin.sh
-conntype SOAP -host wtsc04.itso.ibm.com -port 12002 -username WPADMIN
-password wsadmin
WASX7209I: Connected to process "dmgr" on node WPDmNode using SOAP
connector;  The type of process is: DeploymentManager
WASX7029I: For help, enter: "$Help help"
wsadmin>
```

If the SSL connection cannot be established, it is most likely because the initiator does not trust the server, in this case, the deployment manager. The same problem occurs when visiting a Web site with a browser and the certificate is either not signed by a trusted CA, the certificate is expired, or the certificate name does not match the host name. In contrast to z/OS, the browser will ask the user to accept this certificate and store it in the truststore. The equivalent truststore in z/OS is the keyring, and the certificates have to be connected to the keyring by the RACF administrator.

The CR of the deployment manager is responsible for the connection handling incoming SOAP requests and is the endpoint for the SSL connection. Consequently, the personal server certificate of the deployment manager CR is used for the SSL connections. This certificate is connected to the corresponding keyring of the deployment manager CR (target keyring: `WJkeyring`), which is defined as a keystore in the SSL section of the administrative console. Usually, the `CellDefaultKeystore` is used for the deployment manager and points to the corresponding keyring. This keyring is assigned to the CR user ID of the deployment manager.

In order to establish an SSL connection, the initiator has to trust the personal server certificate of the deployment manager CR. Therefore, the public CA certificate that signed the personal server certificate needs to be added to the initiator's keyring. In order to determine the correct keyring for the initiator, the source node has to be identified.

In this example, the keyring used for the source node depends on the location of the shell script executed by the OMVS user ID. If **`addNode.sh`** and **`wsadmin.sh`** are executed on node `WPNodeA`, the keyring of `WPNodeA` is used, in this case, `WPKeyring`. If **`wsadmin.sh`** is executed on the deployment manager node, the keyring of node `WPDmNode` is used, in this case, `WJKeyring`. The keyring of the source node is specified in the corresponding `ssl.client.props` file, which is located in the following configuration path of that node: `<`*`WAS_HOME`*`>/profiles/default/properties`.

Example 13-2 shows an extract of the `ssl.client.props` file of node `WPNodeA`, which specifies `WPKeyring` as the ClientDefaultKeyStore.

Example 13-2 Extract of the ssl.client.props file of node WPNODEA

```
# KeyStore information
com.ibm.ssl.keyStoreName=ClientDefaultKeyStore
com.ibm.ssl.keyStore=safkeyring:///WPKeyring
com.ibm.ssl.keyStorePassword={xor}Lz4sLCgwLTs=
com.ibm.ssl.keyStoreType=JCERACFKS
com.ibm.ssl.keyStoreProvider=IBMJCE
com.ibm.ssl.keyStoreFileBased=false

# TrustStore information
com.ibm.ssl.trustStoreName=ClientDefaultTrustStore
com.ibm.ssl.trustStore=safkeyring:///WPKeyring
com.ibm.ssl.trustStorePassword={xor}Lz4sLCgwLTs=
com.ibm.ssl.trustStoreType=JCERACFKS
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=false
com.ibm.ssl.trustStoreReadOnly=false
```

This keyring, in conjunction with the initiator's user ID `JVESER`, is used when executing **`addNode.sh`** on node `WPNodeA`. The `WebSphereCA` certificate needs to be added to this keyring to establish an SSL connection with the deployment manager CR as illustrated in Figure 13-3 on page 485.

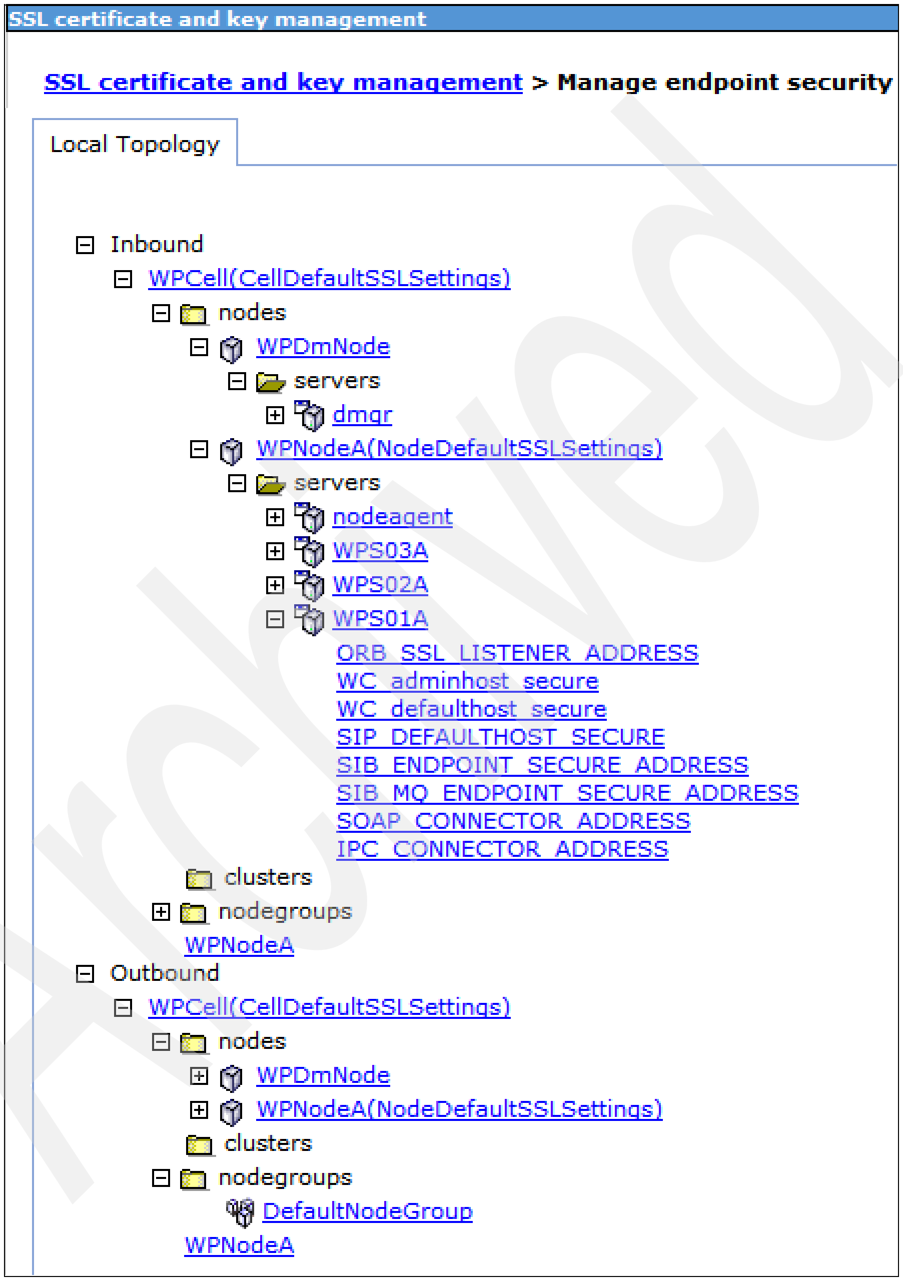
Within a cell, it is common practice to use the same keyring name. This example with two separate keyring names has been chosen to apply this concept also to cross-cell considerations.

In summary, an SSL connection can be established if the signer CA certificate of the target server certificate is added to the initiator's keyring, which depends on the source node.

13.2 Centrally managed SSL

WebSphere provides a approach to organizing SSL configurations by centralizing all SSL configurations on one panel in the administrative console. SSL configurations can be scoped at the cell (Network Deployment only), node, server, and endpoint level so that an alias is not required for every endpoint. The panel is broken into two hierarchies: one tree for inbound endpoints and one tree for outbound endpoints. Select **SSL certificate and key management** → **Manage endpoint security configurations** to see this panel.

A new WebSphere base installation only has one SSL configuration defined at the node level, as illustrated in Figure 13-4.


SSL certificate and key management

SSL certificate and key management > Manage endpoint security

Local Topology

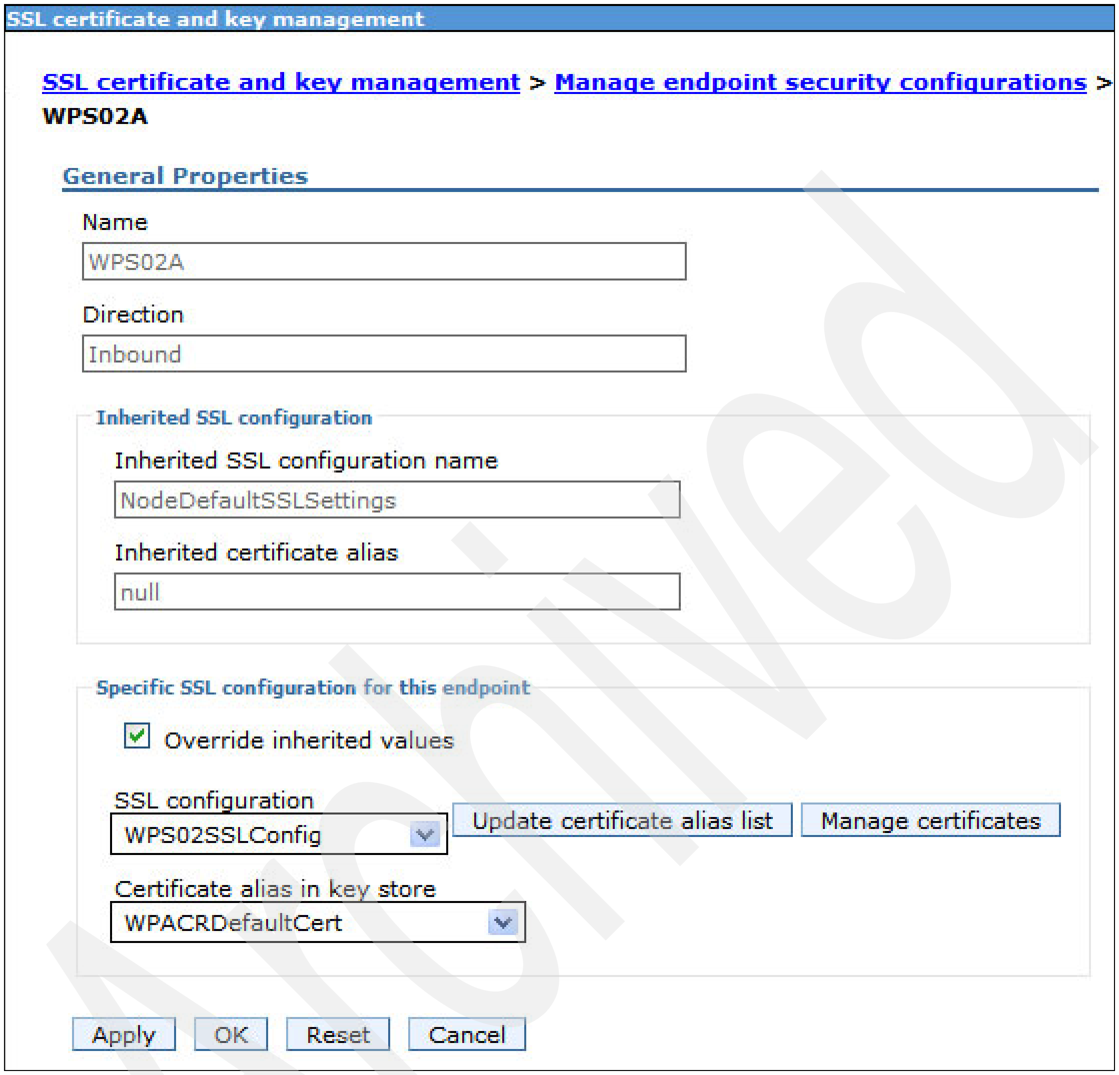
- Inbound
  - WPCell(CellDefaultSSLSettings)
    - nodes
      - WPDmNode
        - servers
          - dmgr
      - WPNodeA(NodeDefaultSSLSettings)
        - servers
          - nodeagent
          - WPS03A
          - WPS02A
          - WPS01A
            - ORB_SSL_LISTENER_ADDRESS
            - WC_adminhost_secure
            - WC_defaulthost_secure
            - SIP_DEFAULTHOST_SECURE
            - SIB_ENDPOINT_SECURE_ADDRESS
            - SIB_MQ_ENDPOINT_SECURE_ADDRESS
            - SOAP_CONNECTOR_ADDRESS
            - IPC_CONNECTOR_ADDRESS
    - clusters
    - nodegroups
    - WPNodeA
- Outbound
  - WPCell(CellDefaultSSLSettings)
    - nodes
      - WPDmNode
      - WPNodeA(NodeDefaultSSLSettings)
    - clusters
    - nodegroups
      - DefaultNodeGroup
    - WPNodeA

Figure 13-4 Centrally managed SSL configuration

In Figure 13-4 on page 488, the servers and endpoints under the inbound topology for `WPNodeA` inherit their SSL configuration NodeDefaultSSLSettings from the node-level SSL configuration. Additional SSL configurations can be defined at a lower scope, overriding the node-level configuration by simply clicking the server or endpoint and choosing a new SSL configuration.

Following any of the paths for the endpoints in Table 13-1 on page 492, the transport chain can be changed from using an SSL configuration alias that is specific to an endpoint to using the centrally configured SSL configuration.

Figure 13-5 on page 490 shows an example of how the SSL inbound channel of a specific application server was changed from using an SSL configuration alias to being centrally managed.



SSL certificate and key management

SSL certificate and key management > Manage endpoint security configurations > WPS02A

General Properties

Name
WPS02A

Direction
Inbound

Inherited SSL configuration

Inherited SSL configuration name
NodeDefaultSSLSettings

Inherited certificate alias
null

Specific SSL configuration for this endpoint

[✓] Override inherited values

SSL configuration
WPS02SSLConfig

Certificate alias in key store
WPACRDefaultCert

Figure 13-5 SSL Configuration: Overwrite the default settings

13.3 WebSphere V7 for z/OS SSSL to JSSE changes

Since WebSphere V6.1, there have been improvements to the way SSL is configured and managed in the administrative console. In previous versions of WebSphere, an SSL repertoire was defined as one of two types, either as Java Secure Socket Extension (JSSE) or System SSL (SSSL). JSSE and SSSL differed in the following ways:

- JSSE:
  - Used Java APIs for SSL
  - Defined with path to a hierarchical file system (HFS) Java keystore file
  - Defined with a path to a safkeyring URI
- SSSL:
  - Used z/OS set of C/C++ APIs for SSL
  - Defined with z/OS SAF keyring name only

Certain endpoints were designed to only work with SSSL or JSSE, and each endpoint needed to be configured with a repertoire alias. It was important to know the location in the administrative console for updating the alias of a transport.

In WebSphere V7, all endpoints with the exception of the daemon use JSSE as the SSL type. The daemon is a lightweight address space that uses system SSL, because the address space does have a Java virtual machine (JVM). All other endpoints are used within the CRs of the deployment manager, node agent, or application server, each equipped with their own JVM.

Table 13-1 on page 492 shows the WebSphere V7 location for specifying an SSL configuration for various endpoints.

Table 13-1 WebSphere V7.0 endpoint details

WebSphere V7.0 for z/OS base server			
Endpoint	**Type**	**Central/alias**	**Administrative console location**
HTTPS	JSSE	Centrally configured (preferred)	**SSL certificate and key management → Manage endpoint security configurations**
		Alias	**Application servers → *server name* → Web container → Web container transport chains → *Choose SSL enabled transport SSL inbound channel (SSL_2)***
RMI-IIOP CSIV2	JSSE	Centrally configured (preferred)	**SSL certificate and key management → Manage endpoint security configurations**
		Alias	**Global security → RMI/IIOP security → CSIv2 inbound/outbound communications**
Daemon SSL	SSSL	Not centrally configured	See Alias
		Alias	**Application servers → *server name* → z/OS location service daemon**
JMX Soap Connectors	JSSE	Centrally configured (preferred)	**SSL certificate and key management → Manage endpoint security configurations**
		Alias	**Application servers → *server name* → Administration Services → JMX connectors → SOAPConnector → Custom Properties → sslConfig**

13.4 Writable SAF keyrings

In previous versions of WebSphere Application Server, you were only allowed to use keystores and truststores that pointed to a SAF keyring in read-only mode. You were not allowed to create, delete, import, or export certificates stored in SAF. These operations needed to be performed by the SAF administrator. In WebSphere V6.1, certificates stored in SAF can be viewed and their expirations can be monitored in the administrative console.

With WebSphere Application Server for z/OS V7, the *writable keyring support* has been introduced. This feature enhances the capabilities of the administrative console to consistently manage keyrings and certificates stored in SAF. With the writable keyring support, certificates can be created and signed by a CA, connected to keyrings, removed from keyrings, imported, exported, and renewed using the administrative console. This major enhancement increases the efficiency of SSL management within WebSphere Application Server for z/OS while SAF still has control over all certificates and keyrings.

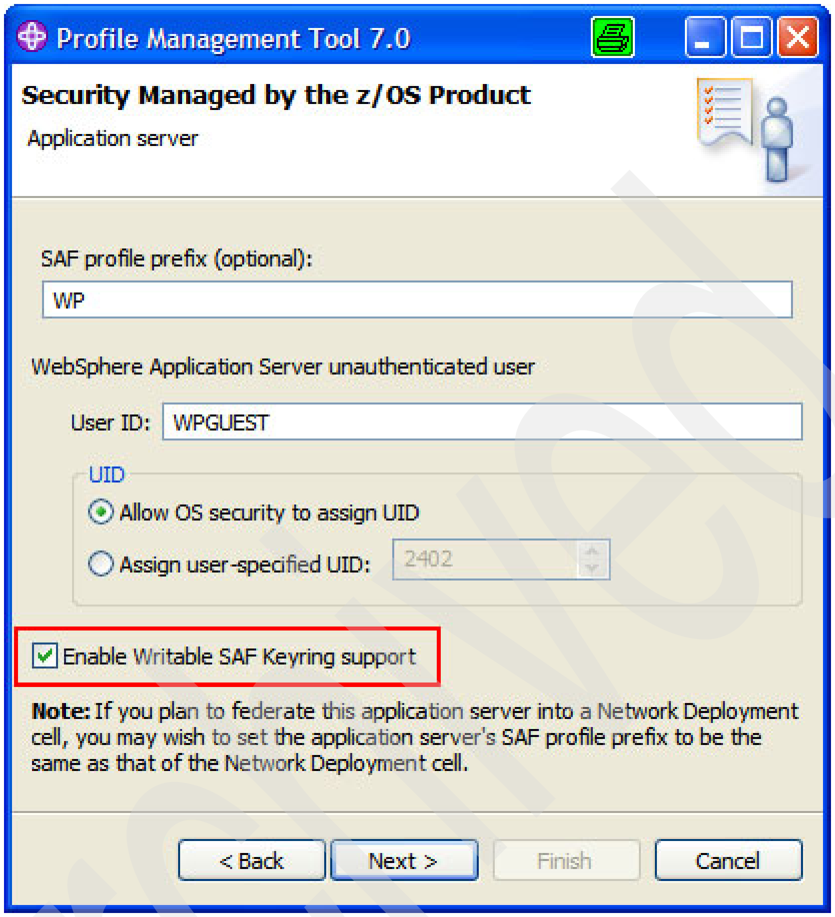
Important: All certificates created with the writable keyring support are generated and signed by Java code and not by SAF. In this case, the writable keyring support only uses SAF to store the generated certificates.

The writable keyring support is completely optional. New keystores and truststores marked as read-only can be created independently from the writable keyring support. When using the read-only JCERACFKS and JCECCARACFS keystores, the certificates in the appropriate SAF keyring can still be viewed in the administrative console.

13.4.1 Prerequisites for writable keyring support

Writable keyring support works with z/OS V1.8 and V1.9, if the required APAR OA22287 (RACF) and APAR OA22295 (SAF) are applied. For z/OS 1.10 and later, these APARs are already included.

In order to make use of the writable keyring support, certain RACF definitions are required. These definitions can be generated automatically with the Profile Management Tool (PMT) during the customization process. During the profile creation of the deployment manager or application server, the z/OS security product must be chosen. In the Security Managed by the z/OS Product panel that is shown in Figure 13-6 on page 494, the option **Enable Writable SAF Keyring support** must be checked.



Profile Management Tool 7.0

Security Managed by the z/OS Product

Application server

SAF profile prefix (optional): WP

WebSphere Application Server unauthenticated user

User ID: WPGUEST

UID

(●) Allow OS security to assign UID

( ) Assign user-specified UID: 2402

[✓] Enable Writable SAF Keyring support

Note: If you plan to federate this application server into a Network Deployment cell, you may wish to set the application server's SAF profile prefix to be the same as that of the Network Deployment cell.

< Back | Next > | Finish | Cancel

Figure 13-6 Enable Writable SAF Keyring support during customization

As a result, the generated BBOxBRAC member in dataset *hlq*.DATA will contain the RACF definitions for the writable keyring support. In addition, this script contains all required keyrings and server certificates.

Required SAF profiles for writable keyring support

In an existing or migrated cell, the necessary RACF definitions need to be created manually by using the following command structures. The writable keyring support makes use of the FACILITY class for global profile checking and the RDATALIB class for ring-specific profile checking. Global profile checking applies to all the key rings whereas ring-specific profile checking applies to a specific keyring. As long as there is no ring-specific profile defined that matches the criteria, the global profile will be used.

The FACILITY profile is structured like the following example:

```
IRR.DIGTCERT.<function>
```

For more details about the IRR.DIGTCERT profile and the RDATALIB class profiles, refer to *z/OS V1R9.0 Security Server RACF Callable Service*, SA22-7691-11, which provides a complete list of RACF commands and options regarding writable keyring support:

http://publibz.boulder.ibm.com/epubs/pdf/ichzd180.pdf

To use the ring-specific profile checking, the RDATALIB class must be activated and on the RACLIST:

```
SETR CLASSACT(RDATALIB)
SETR RACLIST(RDATALIB) GENERIC(RDATALIB)
```

The RDATALIB profile is used to control the access to the keyrings, which consists of the keyring owner, as well as the keyring name, as shown in the following examples. The *<ringowner>* must be in uppercase whereas the *<keyringName>* will automatically formatted to uppercase during profile check:

```
<ringowner>.<keyringName>.LST
<ringowner>.<keyringName>.UPD
```

Depending on the keyring operation, the user ID of the application server started task, which executes the operation, needs to be granted to a LST profile or a UPD profile with READ, UPDATE, or CONTROL permission.

The following RACF commands define two LST profiles for the ring owner *<UID_CR>* and *<UID_SR>*:

```
RDEFINE RDATALIB <UID_CR>.**.LST UACC(NONE)
RDEFINE RDATALIB <UID_SR>.**.LST UACC(NONE)
```

For instance, the READ access to an LST profile allows the granted user ID to list the certificates belonging to this keyring:

```
PERMIT  <UID_CR>.**.LST CLASS(RDATALIB) ID(<UID_CR>) ACC(CONTROL)
PERMIT  <UID_SR>.**.LST CLASS(RDATALIB) ID(<UID_CR>) ACC(CONTROL)
PERMIT  <UID_SR>.**.LST CLASS(RDATALIB) ID(<UID_SR>) ACC(CONTROL)
```

The WebSphere configuration group gets READ access to the keyrings of the application server CR:

```
PERMIT  <UID_CR>.**.LST CLASS(RDATALIB) ID(<GID_CFG>) ACC(READ)
```

If is a user ID is permitted to access a UPD profile with READ access, it can remove all of the certificates from the defined keyring and it can connect its own certificates to the keyring for personal use. With an UPDATE access to this profile, a certificate, which is owned by another user ID, can be added to the keyring for personal use. The CONTROL access is needed to connect a certificate with a private key that is owned by another user ID.

The application server CR gets CONTROL access to the keyrings of the SR, because the keyring operation is always executed by the CR:

```
RDEFINE RDATALIB <UID_CR>.**.UPD UACC(NONE)
RDEFINE RDATALIB <UID_SR>.**.UPD UACC(NONE)

PERMIT  <UID_CR>.**.UPD CLASS(RDATALIB) ID(<UID_CR>) ACC(CONTROL)
PERMIT  <UID_SR>.**.UPD CLASS(RDATALIB) ID(<UID_CR>) ACC(CONTROL)
```

Finally, the RDATALIB class need to be refreshed in order to activate the changes:

```
SETR RACLIST(RDATALIB) REFRESH
```

Important: Currently, *APAR PK81536* is an open APAR regarding writable keyring support with target inclusion for Release 7.0.0.5 of WebSphere Application Server for z/OS. We recommend that you apply this APAR for using writable keyrings. Otherwise, the functionality can be limited after node federation.

Additional keyrings for writable keyring support

In order to create a chained certificate using writable keyring support, a CA root certificate is required. This CA certificate is used to sign new server certificates. If global security and writable keyring support have been enabled during the installation process, new additional keyrings `<keyring_name>.Root` and `<keyring_name>.Signers` have been created.

In addition, the generated CA root certificate has been connected to the Root keyring for personal use. Only CA root certificates that are connected to the Root keyring can sign server certificates using the writable keyring support. The Root keyring correlates to the JCERACFKS keystore DmgrDefaultRootStore or NodeDefaultRootStore, which is located in the following path of the administrative console: **SSL certificate and key management** → **Key stores and certificates**. The keystore usage **Root certificates keystore** must be selected.

This example shows how to add the CA root certificate to the Root keyring for personal use:

```
RACDCERT ID(WPDMGR) CONNECT (RING(WPKeyring.Root) LABEL('WebSphereCA')
CERTAUTH USAGE(PERSONAL))
```

In contrast, certificates connected to the Signers keyring for CERTAUTH use are used as default signer certificates and will be added to any new keystore that is created in the administrative console.

The following example shows how to add a default signer certificate to the corresponding keyring:

```
RACDCERT ID(WPDMGR) CONNECT (RING(WPKeyring.Signers)
LABEL('WebSphereCA') CERTAUTH)
```

The Signers keyring correlates to the JCERACFKS keystore DmgrDefaultSignerStore or NodeDefaultSignerStore, which is located in the following path of the administrative console: **SSL certificate and key management** → **Key stores and certificates**. The keystore usage **Default signers keystore** must be selected.

Important: If global security has been disabled during the customization, these keystores will be file-based and must be manually changed to the JCERACFKS keystores.

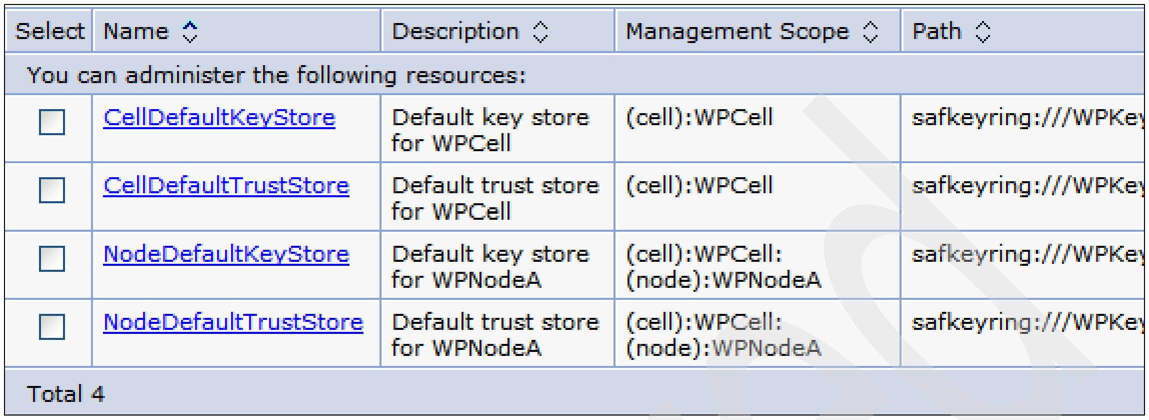
13.4.2 Integration into the administrative console

If global security is enabled during profile creation, the following default keystores have been defined in a cell with a deployment manager and a federated node:

- The CellDefaultKeyStore is defined at the cell scope and includes the keyrings of the deployment manager CRs and SRs.
- The CellDefaultTrustStore is at the cell scope.
- The NodeDefaultKeyStore is defined at the node level and contains the keyrings of the application server CRs and SRs.
- The NodeDefaultTrustStore is at the node level.

Because a RACF keyring can contain trusted as well as personal certificates, there is no difference between the settings of CellDefaultKeystore and CellDefaultTruststore. If writable keyring support has been enabled during the customization, you can use these default keystores to modify keyrings.

You can see these keystores and truststores by selecting **Security** → **SSL certificate and key management** → **Key stores and certificates** (Figure 13-7).

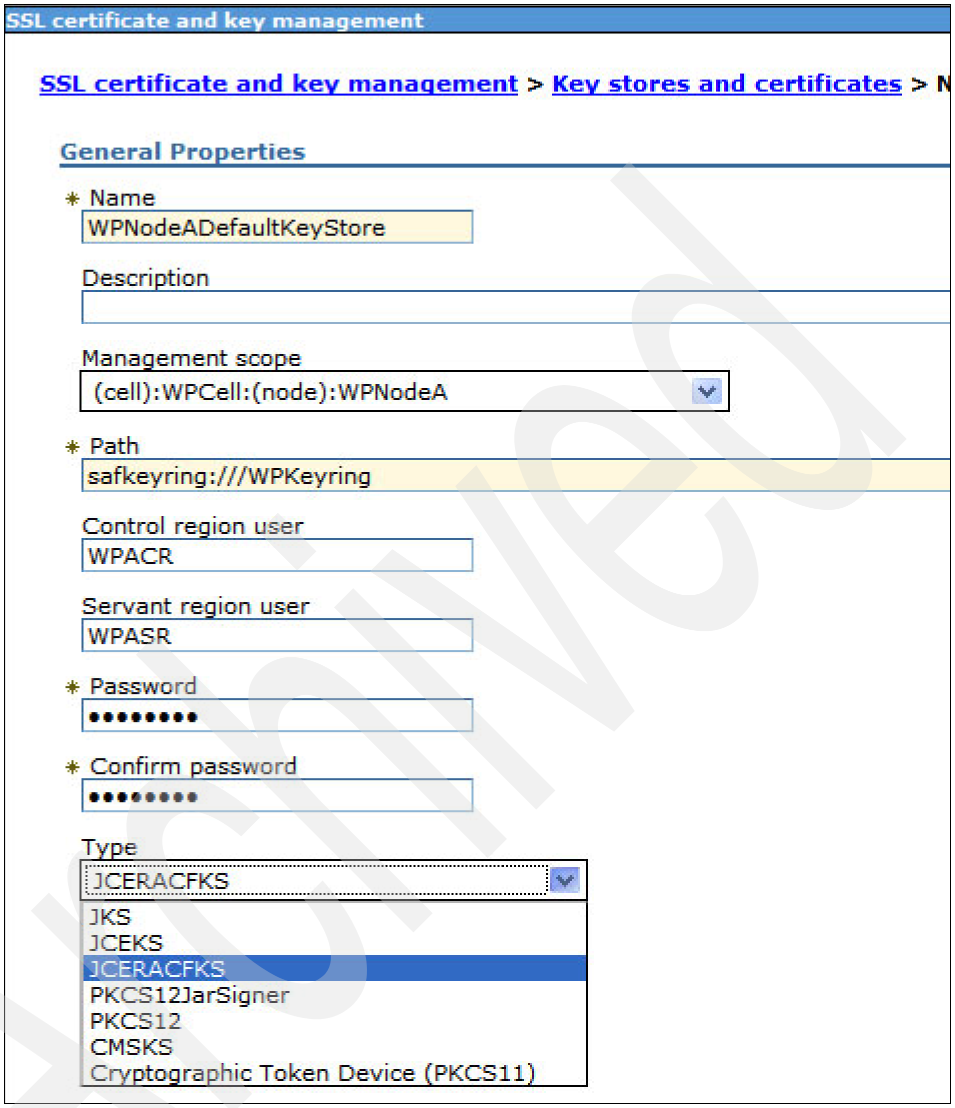


Select	Name	Description	Management Scope	Path
You can administer the following resources:				
☐	CellDefaultKeyStore	Default key store for WPCell	(cell):WPCell	safkeyring:///WPKe
☐	CellDefaultTrustStore	Default trust store for WPCell	(cell):WPCell	safkeyring:///WPKe
☐	NodeDefaultKeyStore	Default key store for WPNodeA	(cell):WPCell:(node):WPNodeA	safkeyring:///WPKe
☐	NodeDefaultTrustStore	Default trust store for WPNodeA	(cell):WPCell:(node):WPNodeA	safkeyring:///WPKe
Total 4				

Figure 13-7 Default key and trust stores

Create a new keystore

You can create a new keystore by using the following path in the administrative console: **Security** → **SSL certificate and key management** → **Key stores and certificates** → **New** (Figure 13-8 on page 499).


SSL certificate and key management

SSL certificate and key management > Key stores and certificates > N

General Properties

* Name
WPNodeADefaultKeyStore

Description

Management scope
(cell):WPCell:(node):WPNodeA

* Path
safkeyring:///WPKeyring

Control region user
WPACR

Servant region user
WPASR

* Password
●●●●●●●●

* Confirm password
●●●●●●●●

Type
JCERACFKS

JKS
JCEKS
JCERACFKS
PKCS12JarSigner
PKCS12
CMSKS
Cryptographic Token Device (PKCS11)

Figure 13-8 Creating a new JCERACFKS keystore using safkeyring

A *keyring* always consists of a keyring name and a user ID. Prior to Version 7.0, the administrative console, from a keyring management perspective, was unable to distinguish between the CR and the SR keyrings. The certificate administration was limited to reading certificates from a single keystore object that represented both the SR and CR keyrings. In V7, additional fields have been added for the writable keyring support that create a new keystore, the CR user, and the SR user. If the optional fields for the CR user ID and SR user ID are not filled out, the writable keyring support is not used. Figure 13-8 on page 499 is an example of the values that are used to create a new keystore. To use writable keyring support, the bold marked values must be used:

- Name: *`keystore_name`*
- Management scope: *`node`*
- Path: **`safkeyring:///`**`WPKeyring`
- CR user: *`Control region user ID`*
- SR user: *`Servant region user ID`*
- Password: ***`password`***
- Confirm password: ***`password`***
- Type: **JCERACFKS**
- Clear the check mark from the **Read Only** check box for using writable keyring support (This setting is in the same panel beneath the Type field, but it is not shown in Figure 13-8 on page 499).

The panel that is shown in Figure 13-8 on page 499 is used to accommodate file-based and SAF-based keystore types. *File-based keystores* are normally protected with a password when the keystore database is first created, and the administrative console needs to have this password specified to access a file-based keystore.

For certificates stored in RACF, the certificates are protected by the access granted to the user ID by the RACF administrator. The corresponding Java standard JSSE still requires that a password is specified to access a SAF keystore, although the certificates are protected by the user ID's access privileges in the RACF database. The certificates in RACF will not be accessible unless the password is specified correctly in this panel.

Important: The administrative console stores the exact string of the safkeyring path in the `security.xml` file in the UNIX System Services. Make sure that no trailing spaces are used when specifying the safkeyring path and make sure that you use three slashes:

```
safkeyring:/// (with three slashes)
```

The administrative console only confirms that the change password entry and the confirm password entry match. It does not verify that the password value is the valid password that must be used to access certificates using a safkeyring URI. If these fields are not entered correctly, you will not be able to view or access your certificates in RACF, and ports configured with this SSL configuration will not work correctly.

Instead of specifying the CR user ID and SR user ID, you can define a safkeyring path that includes the keyring name and the corresponding user ID:

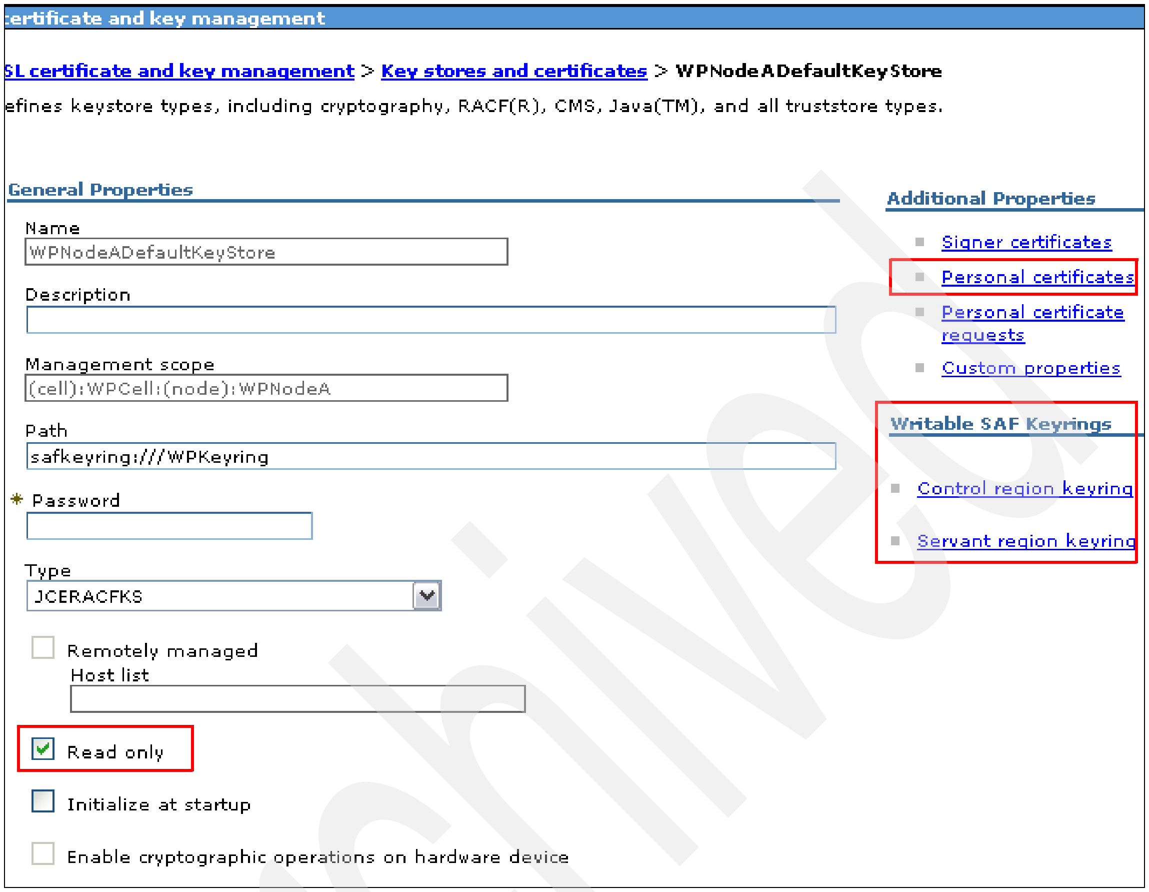
```
safkeyring://UID/keyring_name
```

Compared to the first option with the CR user ID and SR user ID, in this solution, two separate keystores need to be defined to achieve the same aim.

Here is an example for the second option. The optional fields for CR user ID and SR user ID can be left blank:

- Name: `WPNodeADefaultKeystore`
- Management scope: `WPNodeA`
- Path: `safkeyring:`**`//WPDMGR/WPKeyring`**
- Control region user:
- Servant region user:
- Password: ***`password`***
- Confirm password: ***`password`***
- Type: **JCERACFKS**
- Clear the check mark from the **Read Only** check box for using writable keyring support.

After creating the keystore, you can click the new keystore name in the list of keystores and certificates to view it (Figure 13-9 on page 502). At first glance, nothing seems to be changed from the previous version of WebSphere. In fact, the properties page is still the read-only view of the previous version.


ertificate and key management

SL certificate and key management > Key stores and certificates > WPNodeADefaultKeyStore

efines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore types.

General Properties

Name
WPNodeADefaultKeyStore

Description

Management scope
(cell):WPCell:(node):WPNodeA

Path
safkeyring:///WPKeyring

* Password

Type
JCERACFKS

☐ Remotely managed
Host list

☑ Read only

☐ Initialize at startup

☐ Enable cryptographic operations on hardware device

Additional Properties

- Signer certificates
- Personal certificates
- Personal certificate requests
- Custom properties

Writable SAF Keyrings

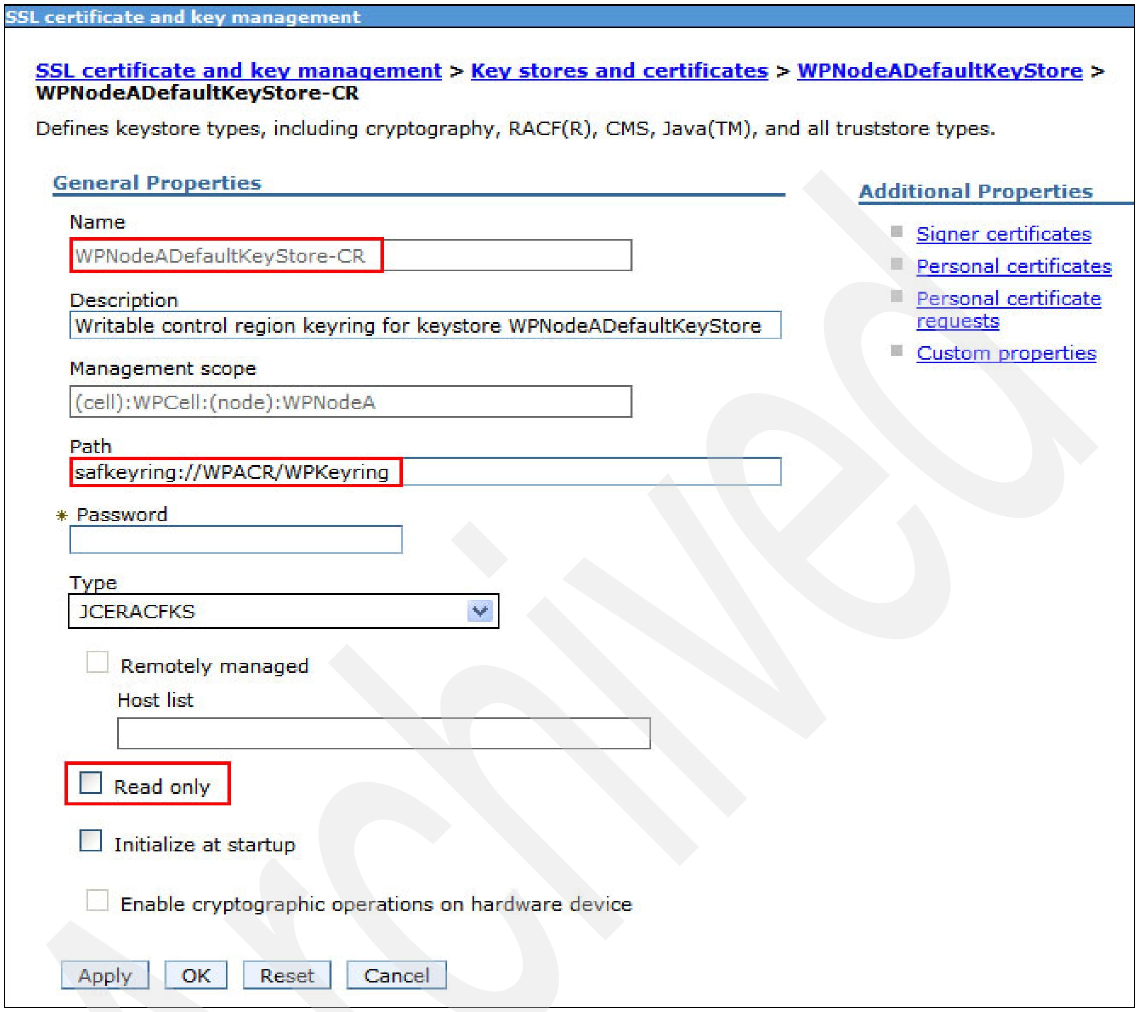
- Control region keyring
- Servant region keyring

Figure 13-9 Viewing the new keystore

You can browse the keyrings of the CR and SR by clicking **Personal certificates**, but as the check box at the bottom indicates, in read-only mode.

For writable keyring support, two additional keystores objects have been created in the right navigation bar for the CR and SR that are called **Control region keyring** and **Servant region keyring** as shown in Figure 13-9.

Select **Control region keyring** to access the provided keyring name with the provided CR user ID (Figure 13-10 on page 503).


SSL certificate and key management

SSL certificate and key management > Key stores and certificates > WPNodeADefaultKeyStore > WPNodeADefaultKeyStore-CR

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore types.

General Properties

Name: WPNodeADefaultKeyStore-CR

Description: Writable control region keyring for keystore WPNodeADefaultKeyStore

Management scope: (cell):WPCell:(node):WPNodeA

Path: safkeyring://WPACR/WPKeyring

* Password:

Type: JCERACFKS

☐ Remotely managed

Host list:

☐ Read only

☐ Initialize at startup

☐ Enable cryptographic operations on hardware device

Figure 13-10 Writable keystore of CR

In Figure 13-10, the settings of the writable CR keystore are displayed:

- The name of the keystore `keystore_name-CR` is the same as the read-only keystore except for the `-CR` suffix, indicating the writable CR keystore.
- The safkeyring path (URI) has changed to `safkeyring://UID_CR/keyring_name`. This URI specifies the keyring name as well as the corresponding user ID.
- Finally, the Read only flag is *not* selected.

In the SR keystore, these values, such as the keystore name `keystore_name-SR`, are adjusted to the SR.

Under Additional properties on the right navigation panel, click **Personal certificates**, and the certificates connected to the corresponding keyring are displayed. Click an individual certificate to see detailed information about it.

Important: Only certificates with the status `TRUST` can be viewed in the administrative console.

wsadmin commands

The following **`wsadmin`** command displays all of the certificates that belong to this keystore:

```
AdminTask.listPersonalCertificates('[-keyStoreName
NodeDefaultKeyStore-CR -keyStoreScope (cell):WPCell:(node):WPNodeA ]')
```

The following command displays details about a specific certificate:

```
AdminTask.getCertificateChain('[-certificateAlias
DefaultWASCert.WPNODEA -keyStoreName NodeDefaultKeyStore-CR
-keyStoreScope (cell):WPCell:(node):WPNodeA ]')
```

The following **`AdminTask`** command creates a new writable keystore:

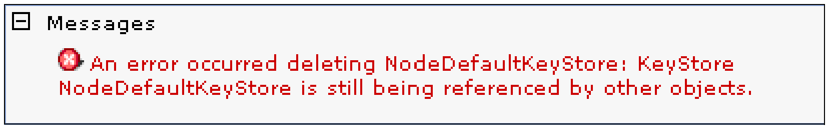
```
AdminTask.createKeyStore('[-keyStoreName <keystore_name> -scopeName
(cell):WPCell:(node):WPNodeA -keyStoreDescription -keyStoreLocation
safkeyring:///<keyring_name> -keyStorePassword password
-keyStorePasswordVerify password -keyStoreType JCERACFKS
-keyStoreInitAtStartup false -keyStoreReadOnly false -keyStoreStashFile
false -keyStoreUsage SSLKeys -controlRegionUser <UID_CR>
-servantRegionUser <UID_SR>]')
```

The existing JCERACFKS without writable keyring support can be enabled by using the following command:

```
AdminTask.enableWritableKeyrings('[-keyStoreName <keystore_name>
-scopeName (cell):WPCell:(node):WPNodeA -controlRegionUser <UID_CR>
-servantRegionUser <UID_SR>]')
```

Deleting keystores and truststores

Keystores and truststores can be deleted if no SSL configurations reference them. The following example error message (Figure 13-11 on page 505) appears when attempting to delete a keystore or truststore that is still being used by an SSL configuration.


Messages

An error occurred deleting NodeDefaultKeyStore: KeyStore NodeDefaultKeyStore is still being referenced by other objects.

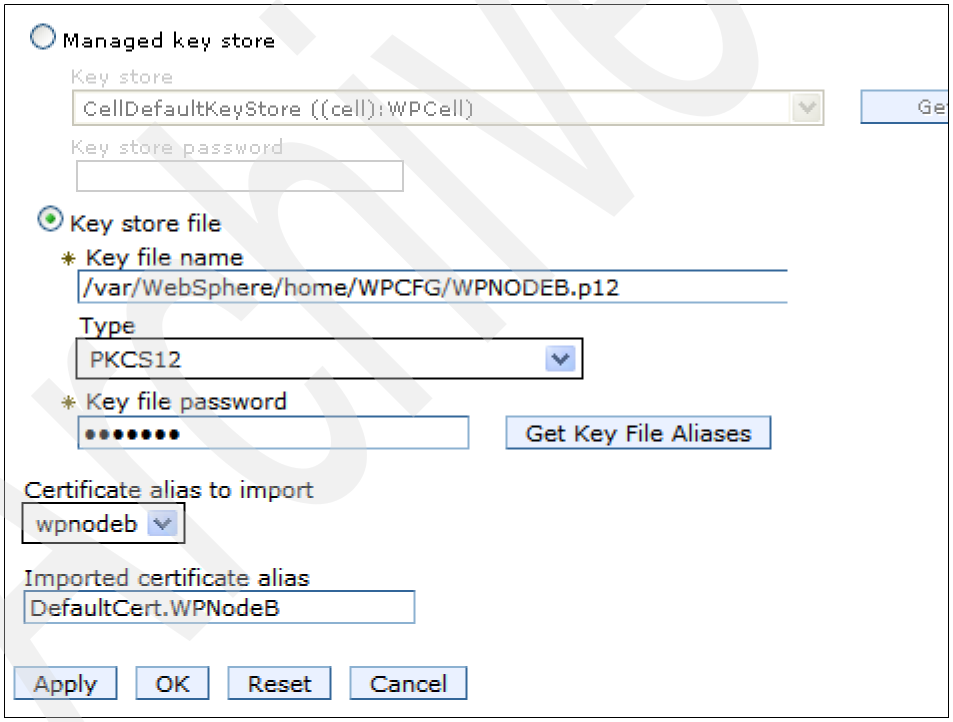
Figure 13-11 Error received when deleting a keystore defined in an SSL configuration

To resolve this issue, delete the SSL configuration that has this keystore specified first before deleting the keystore.

13.4.3 Importing personal certificates

You can import personal certificates into SAF by following these steps:

1. In the Personal certificates panel of the writable keystore, select a certificate to be imported, click **Import**, and complete the required fields (Figure 13-12).



Key store file

Key file name: /var/WebSphere/home/WPCFG/WPNODEB.p12

Type: PKCS12

Key file password: ●●●●●●●

Certificate alias to import: wpnodeb

Imported certificate alias: DefaultCert.WPNodeB

Figure 13-12 Importing a external certificate into the keyring

– Select **Key store file** to import a certificate from the UNIX System Services file system, as shown in this example. The alternative is to select

Managed key store to import a certificate from another WebSphere keystore. If a RACF keyring is selected, the password is `password`.

– Key file name:

  Enter the complete path to the certificate in the UNIX System Services in the Key file name field. Up-front, the certificate must be uploaded to a UNIX System Services path using a secured FTP connection. Ensure that the started task has at least read access to the selected folder.

– Type:

  Select the type of certificate to be imported. In this case, the certificate is packaged in the `PKCS12` format.

– Key file password:

  The password that was used to encrypt this certificate.

– Certificate alias to import

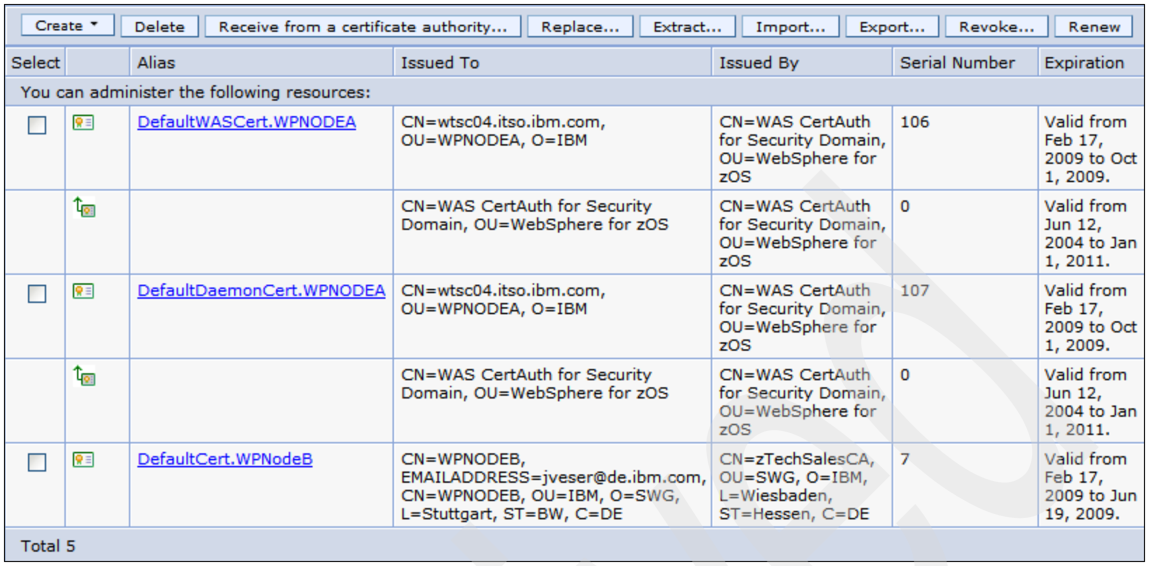
  Click **Get Key File Alias** and the appropriate certificate alias drop-down list will be updated. Select the certificate alias to be imported.

– Imported certificate alias

  This field specifies the new alias name of the certificate in RACF. Usually, this name matches the selected certificate alias.

2. Click **OK** to import the certificate, and save the changes.

The imported certificate will be displayed in the keystore as shown in Figure 13-13 on page 507. The certificate has been imported to RACF and connected to the selected keyring and user ID. Click the imported certificate to see the details.



Select		Alias	Issued To	Issued By	Serial Number	Expiration
You can administer the following resources:						
☐		DefaultWASCert.WPNODEA	CN=wtsc04.itso.ibm.com, OU=WPNODEA, O=IBM	CN=WAS CertAuth for Security Domain, OU=WebSphere for zOS	106	Valid from Feb 17, 2009 to Oct 1, 2009.
			CN=WAS CertAuth for Security Domain, OU=WebSphere for zOS	CN=WAS CertAuth for Security Domain, OU=WebSphere for zOS	0	Valid from Jun 12, 2004 to Jan 1, 2011.
☐		DefaultDaemonCert.WPNODEA	CN=wtsc04.itso.ibm.com, OU=WPNODEA, O=IBM	CN=WAS CertAuth for Security Domain, OU=WebSphere for zOS	107	Valid from Feb 17, 2009 to Oct 1, 2009.
			CN=WAS CertAuth for Security Domain, OU=WebSphere for zOS	CN=WAS CertAuth for Security Domain, OU=WebSphere for zOS	0	Valid from Jun 12, 2004 to Jan 1, 2011.
☐		DefaultCert.WPNodeB	CN=WPNODEB, EMAILADDRESS=jveser@de.ibm.com, CN=WPNODEB, OU=IBM, O=SWG, L=Stuttgart, ST=BW, C=DE	CN=zTechSalesCA, OU=SWG, O=IBM, L=Wiesbaden, ST=Hessen, C=DE	7	Valid from Feb 17, 2009 to Jun 19, 2009.
Total 5						

Figure 13-13 Imported certificate in the keyring

Important: During the import and export of certificates to and from managed SAF keystores, if the certificate already exists in SAF under another label, it will be connected to the keyring with the existing label regardless of the label assigned to the certificate.

13.4.4 Exporting personal certificates

To export a personal certificate:

1. In the Personal certificates panel of the writable keystore, select a certificate to be exported and click **Export**.
2. Specify the password in the Key store password field.
3. The alias needs to be the certificate name.
4. A certificate can either be exported to another truststore, including SAF keyrings, or to a file in the UNIX System Services:
   a. To export to another keystore, select **Managed key store** and specify the target keystore.
   b. To export to the UNIX System Services file system, select **Key store file**. Specify the file name, including the complete UNIX System Services path and the export type. In most cases, `PKCS12` is used as an export format. The certificate will be encrypted with the specified password.

The corresponding **wsadmin** command to step 4b is:

```
AdminTask.exportCertificate('[-keyFilePath
/var/WebSphere/home/WPCFG/DefaultWASCert-WPNODA.p12 -keyFilePassword
passwd -keyFileType PKCS12 -keyStorePassword password -certificateAlias
DefaultWASCert.WPNODEA -aliasInKeyStore DefaultWASCert.WPNODEA
-keyStoreName NodeDefaultKeyStore-CR -keyStoreScope
(cell):WPCell:(node):WPNodeA ]')
```

13.4.5 Creating personal certificates

When using the writable keyring support, you can create three types of certificates:

- Self-signed certificates

  Self-signed certificates are normally created in the DmgrDefaultRootStore or NodeDefaultRootStore keystores so that new server certificates can be signed by this self-signed certificate.

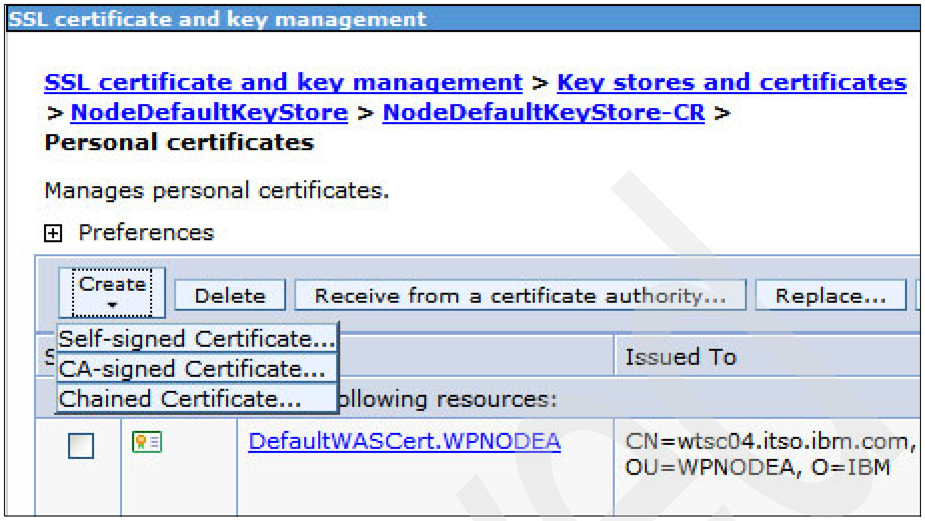
- CA-signed certificates

  With the CA-signed certificate option, certificate requests to an external CA can be issued and the received certificate will be stored in the chosen keyring. As a prerequisite for this function, a CA client has to be created, including a specific implementation class for the external CA.

- Chained certificates

  The chained certificate option allows you to create a new personal server certificate that is signed by a CA from the DmgrDefaultRootStore keystore and attached to the selected keyring.

To create a new personal server certificate, select **Security** → **SSL certificate and key management** → **Key stores and certificates** → ***key_store_name*** → **Control region keyring** → **Personal certificates**. Click **Create** and choose the appropriate certificate type (Figure 13-14 on page 509).



SSL certificate and key management

SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > NodeDefaultKeyStore-CR > Personal certificates

Manages personal certificates.

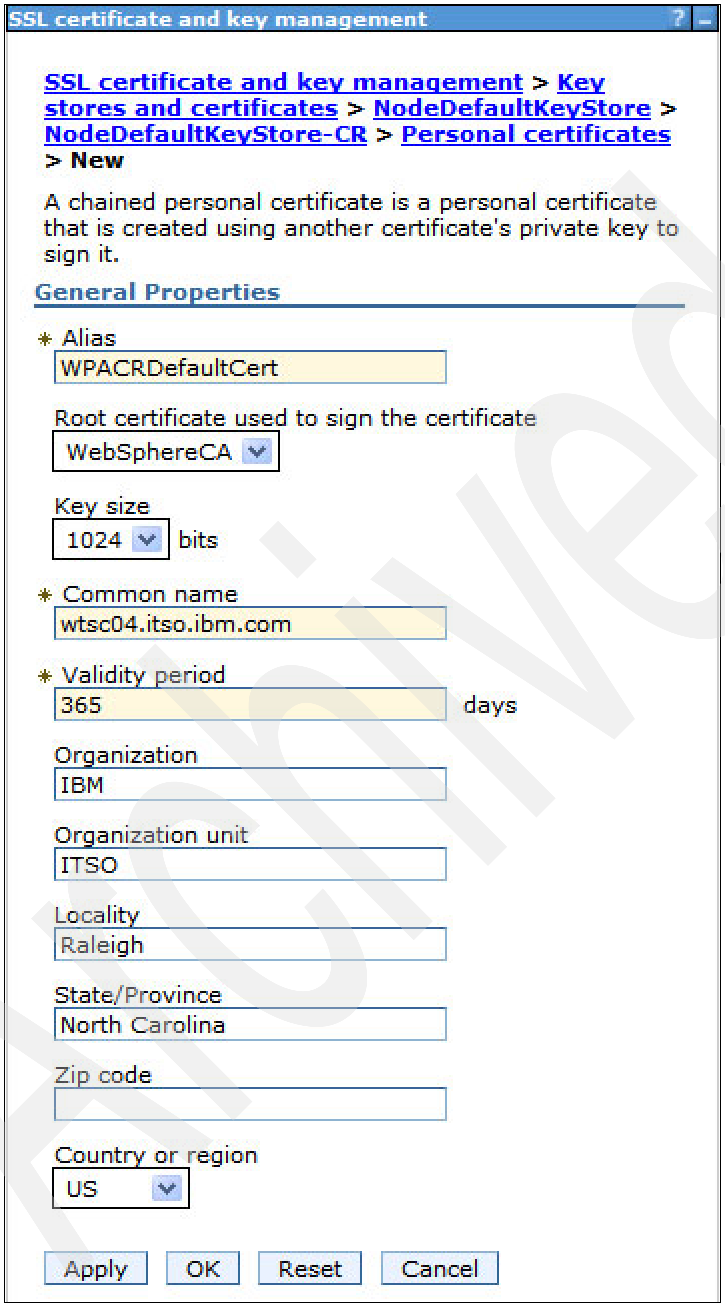
⊞ Preferences

			Issued To
☐		DefaultWASCert.WPNODEA	CN=wtsc04.itso.ibm.com, OU=WPNODEA, O=IBM

Figure 13-14 Create a new personal certificate

The configuration page for the new certificate will open.

For example, to create a chained certificate, you will see the panel in Figure 13-15 on page 510.


SSL certificate and key management

SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > NodeDefaultKeyStore-CR > Personal certificates > New

A chained personal certificate is a personal certificate that is created using another certificate's private key to sign it.

General Properties

* Alias
WPACRDefaultCert

Root certificate used to sign the certificate
WebSphereCA

Key size
1024 bits

* Common name
wtsc04.itso.ibm.com

* Validity period
365 days

Organization
IBM

Organization unit
ITSO

Locality
Raleigh

State/Province
North Carolina

Zip code

Country or region
US

Figure 13-15 Create a new chained certificate

The required information is:

- Alias

  The alias given to this certificate is the same as the label name in RACF and will be displayed in the keyring overview.

- Root certificate used to sign the certificate

  Any CA root certificate assigned to the DmgrDefaultRootStore or NodeDefaultRootStore can be selected. The keystore must point to the keyring `<keyring_name>.Root` as described in "Additional keyrings for writable keyring support" on page 496. Select the CA root certificate to sign the new personal certificate.

- Common name

  Normally, the common name corresponds with the host name of the target server. The client will check if the host name used for the SSL connection matches the common name of the server certificate. If it does not match, the connection can be denied.

- Validity period

  The validity period of the personal server certificate needs to end before the the CA root certificate expiration date.

Complete the remaining values according to your organizational structure. Then, click **OK** and **Save** to create the personal certificate.

The newly created certificate appears in the keyring overview.

The following command displays a personal certificate in RACF that has been created by the administrative console:

```
RACDCERT LIST (label('personal_certificate_label')) ID(user ID)
```

Example 13-3 shows the output of this command.

Example 13-3 List the new personal certificate in RACF

```
Label: WPACRDefaultCert
Certificate ID: 2QXm18HD2ebXwcPZxIWGgaSTo8OFmaNA
Status: TRUST
Start Date: 2009/03/05 18:12:38
End Date:   2010/03/05 18:12:38
Serial Number:
     >11287ED6C7E77277<
Issuer's Name:
    >CN=WAS CertAuth for Security Domain.OU=WebSphere for zOS<
Subject's Name:
     >CN=wtsc04.itso.ibm.com.OU=ITSO.O=IBM.L=Raleigh.SP=North
Carolina.C=US
Subject's AltNames:
  EMail:
ProfileUUID:default-DEPLOYMENT_MANAGER-8dff92c9-a143-4496-8379-502
         bb062479
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
  Ring Owner: WPACR
  Ring:
     >WPKeyring<
```

13.4.6 Removing certificates from a keyring

A certificate can be removed from the keyring by selecting the appropriate certificate and clicking Delete in the certificate overview. The certificate will not be deleted in SAF. If the certificate has accidentally been removed from the keyring, it can be reconnected through SAF or it can be imported from the DmgrDefaultDeletedStore keystore.

Currently in the default configuration, all certificates that have been removed from the keyring will also be stored in the keystore DmgrDefaultDeletedStore in a Network Deployment environment and in the keystore NodeDefaultDeletedStore in a single server environment. These keystores are file-based PKCS12 keystores and are located in the following UNIX System Services path:

`DMGR_Config_Root/cells/cell_name/nodes/dmgr_node/deleted.p12`

This special keystore cannot be substituted by a SAF keyring.

Tip: Certificates, which have been removed from a keyring, are still present in SAF, as well as in the file-based keystore DmgrDefaultDeletedStore. If a mixture of SAF keyrings and file-based keystores does not conform to your enterprise security policy, you can deactivate the functionality of this keystore by marking it as read-only.

In order to display the DmgrDefaultDeletedStore, or equivalent, use the administrative console to select **SSL certificate and key management** → **Key stores and certificates** and then select **Deleted certificates keystore** for keystore usage. Click **DmgrDefaultDeletedStore** and set the read-only flag. Before applying this change, specify the default password for file-based keystores, which is `WebAS`. Click **OK**, and save the changes.

To delete the certificate from SAF, it must be removed by the SAF administrator. To connect a certificate to a keyring or delete a certificate from RACF, you can use the RACDCERT commands as described in 13.5.4, “Deleting certificates and keyrings” on page 518.

The corresponding **wsadmin** task command removes a certificate from a keyring:

```
AdminTask.deleteCertificate('[-keyStoreName NodeDefaultKeyStore-CR
-keyStoreScope (cell):WPCell:(node):WPNodeA -certificateAlias
MyCertificate ]')
```

13.4.7 Renewing certificates

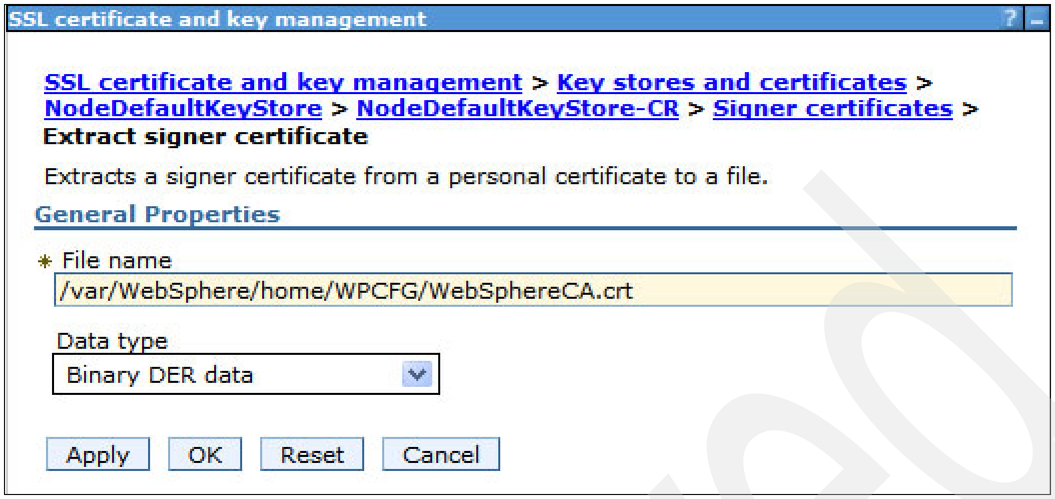
A certificate can be renewed by selecting the certificate and clicking **Renew** in the personal certificates panel. The original certificate is not physically deleted from RACF. Instead, the original certificate is removed from the selected keyring. By renewing a certificate, a new certificate is generated and the alias (label) of the certificate is incremented by appending `_1`, `_2`, and so forth to the existing certificate label.

13.4.8 Exporting public certificates

Public certificates can be exported in the personal certificates panel or in the signer certificates panel.

Select **Security** → **SSL certificate and key management** → **Key stores and certificates** → ***key_store_name*** → **Control region keyring** → **Signer certificates**. Select the certificate, and click **Extract**.

Just specify the file name, including UNIX System Services path and the export format, as shown in Figure 13-16 on page 514.


SSL certificate and key management

SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > NodeDefaultKeyStore-CR > Signer certificates >
Extract signer certificate

Extracts a signer certificate from a personal certificate to a file.

General Properties

* File name
/var/WebSphere/home/WPCFG/WebSphereCA.crt

Data type
Binary DER data

Apply | OK | Reset | Cancel

Figure 13-16 Export a signer certificate

13.4.9 Common error messages

If RACF authority is not granted, you receive the following message when attempting certificate write operations on a keyring:

```
Error Message: An error occurred creating the key store: R_datalib
(IRRSDL00) error: One or more updates could not be completed. Not RACF
authorized to use the requested service. Function code: (7) Return
Codes: (8, 8, 8)
```

If you attempt to create a new keyring or perform a specific certificate write operation and do not have native writable support, you receive the following message:

```
R_datalib (IRRSDL00) error: One or more updates could not be completed.
Requested Function_code not defined. Function code: (7) Return Codes:
(8, 8, 20)
```

13.5 RACF certificate management

This section demonstrates certain administrative tasks with SAF keyrings and includes sample RACF commands. The tasks include viewing, importing, exporting, and deleting certificates.

For more details about the RACF commands, refer to *Security Server RACF Command Language Reference,* SA22-7687, which provides a complete list of RACF security commands and options.

The following RACF commands show how to obtain corresponding certificate information that is displayed in the administrative console:

- View a list of certificates on a keyring for a user ID:

  ```
  RACDCERT LISTRING(keyring_name) ID(user ID)
  ```

- Display all keyrings related to a user ID:

  ```
  RACDCERT LISTRING(*) ID(user ID)
  ```

- View certificate authority information:

  ```
  RACDCERT CERTAUTH LIST(LABEL('certificate_authority_label'))
  ```

- View personal certificate information for a user ID:

  ```
  RACDCERT LIST (label('personal_certificate_label')) ID(user ID)
  ```

- Display all certificates related to a user ID:

  ```
  RACDCERT ID(user ID) LIST
  ```

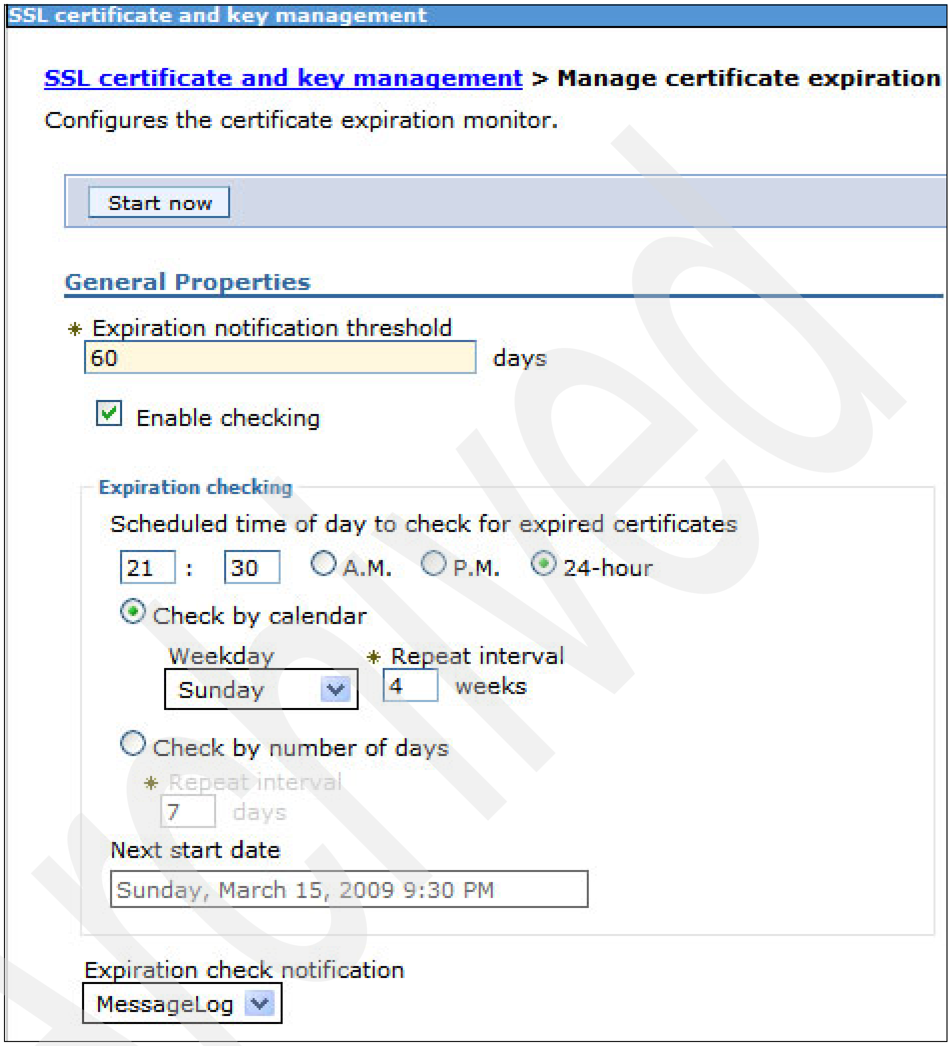
13.5.1 Monitoring certificate expiration

WebSphere Application Server V7 provides a certificate expiration monitoring task that cycles through all the keystores listed in the `security.xml` file to check for expired certificates. The cycle can be specified by the day of the week, or by a certain number of days, and a notification can be provided as a message in the message log, or as an e-mail. For e-mail notification, a Simple Mail Transfer Protocol (SMTP) server address needs to be specified with a recipient's e-mail address. WebSphere Application Server issues an expiration warning before expiration based on the number of days specified in the Expiration notification threshold field.

The administrative console path to manage certificate expiration is **SSL certificate and key management** → **Manage certificate expiration**.

Restriction: At the bottom of this window, there are two additional options for automatic replacement of expiring certificates and for deleting the certificates after replacement. These functions will not work with SAF keyrings even if the writable keyring support is activated for that particular certificate. But monitoring the certificate expiration works with SAF keyrings.

Figure 13-17 illustrates the options for managing certificate expiration.


SSL certificate and key management

SSL certificate and key management > Manage certificate expiration

Configures the certificate expiration monitor.

General Properties

* Expiration notification threshold
60 days

[✓] Enable checking

Expiration checking

Scheduled time of day to check for expired certificates
21 : 30 A.M. P.M. [✓ 24-hour]

[✓ Check by calendar]

Weekday: Sunday
* Repeat interval: 4 weeks

Check by number of days

* Repeat interval: 7 days

Next start date
Sunday, March 15, 2009 9:30 PM

Expiration check notification
MessageLog

Figure 13-17 Certificate management expiration panel

The options to automatically replace expiring self-signed certificates and to delete expiring certificates and signers after replacement are not applicable to certificates stored in RACF. More information is provided in the WebSphere Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_sslcertmonitoring.html

13.5.2 Importing certificates

The following commands show how to import a certificate from an HFS file into a keyring:

- Copy the certificate to a data set from an HFS file:

  ```
  cp -B /tmp/certificate.p12 "//'HLQ.CERT.NAME.P12'"
  ```

  ```
  cp -B /tmp/certificate.cer "//'HLQ.CERT.NAME.CER'"
  ```

- Add the certificate from the data set to RACF with a label:

  ```
  RACDCERT ADD('HLQ.CERT.NAME.P12') withlabel('certificate_label')
  password('password')'

  RACDCERT ADD('HLQ.CERT.NAME.CER') withlabel('certificate_label')
  ```

13.5.3 Exporting certificates

You can use RACF commands to export a certificate from a keyring to a data set, and then the data set can be placed into the HFS as a file to be used on other systems.

The following RACF commands and steps show an example of exporting a signer certificate or personal certificate into an HFS file to be transferred to another system:

- Exporting a signer certificate to an HFS file:

  ```
  RACDCERT ID(user ID) CERTAUTH
  EXPORT(LABEL('certificate_authority_label')) DSN('HLQ.CA.NAME.CRT')
  FORMAT(CERTDER)
  ```

  After the certificate has been exported to a data set in DER format, the data set can be moved to an HFS file with the following command:

  ```
  OPUT 'CA.DSN.NAME' '/tmp/ca.crt' binary convert(no)
  ```

- Exporting a personal certificate to an HFS file:

  ```
  RACDCERT ID(user ID) EXPORT(LABEL('personal_certificate_label'))
  DSN('HLQ.CERT.NAME.P12') FORMAT(PKCS12DER) PASSWORD('passwd')
  ```

  After the certificate has been exported to a data set in PKCS12 format, the data set can be moved to an HFS file with the following command:

  ```
  OPUT 'HLQ.CERT.NAME.P12' '/tmp/personal.p12' binary convert(no)
  ```

Important: While the CERTDER and CERTB64 keywords indicate that only a certificate is exported, the PKCS12DER and PKCS12B64 keywords export the certificate and the private key (which must exist and must not be an Integrated Cryptographic Service Facility (ICSF) key). Be careful when specifying the format to not mistakenly export a private key with the PKCS12 format when your intent is to export the signer certificate only.

Table 13-2 shows the available formats that the WebSphere administrative console can import from a file.

Table 13-2 Export formats for certificates

Format	Description
CERTB64	Specifies X.509 certificate and public key that has been encoded using Base64. This format can be transferred in ASCII.
CERTDER	Specifies encoded X.509 certificate and public key encoded in binary DER format. This format must be transferred in binary.
PKCS12B64	Specifies a PKCS #12 package that has been encoded using Base64. This format might not be compatible with non-IBM applications and must be transferred in ASCII.
PKCS12DER	Specifies a PKCS #12 package encoded using binary DER format. This format must be transferred in binary.

Important: A personal certificate that contains a private key must always be transferred to the client in binary using a *secured* FTP connection. If an unauthorized person gets a copy of this private key, all SSL connections between clients and the server can be decrypted by this person. The concept of SSL is based on the secrecy of the private key. By using unsecured connections for private keys, the security system is extremely vulnerable.

13.5.4 Deleting certificates and keyrings

The certificate delete operation in the administrative console using the writable keyrings removes only the certificate from the keyring, but it does not delete the certificate itself. There might be other keyring references to this certificate, which cannot be displayed by the administrative console.

The following RACF commands remove a certificate from a keyring, remove a keyring, and delete certificates for a user ID:

- Remove a CA signer certificate from a keyring:

```
RACDCERT REMOVE(CERTAUTH LABEL('certificate_authority_label')
RING(keyring_name)) ID(user ID))
```

- Remove a personal certificate from a keyring:

```
RACDCERT REMOVE(LABEL('personal_certificate_label')
RING(keyring_name)) ID(user ID)
```

- Delete the keyring:

```
RACDCERT DELRING(keyring_name) ID(user ID)
```

- Delete a CA signer certificate:

```
RACDCERT CERTAUTH DELETE(LABEL('certificate_authority_label'))
```

- Delete a personal certificate:

```
RACDCERT DELETE(LABEL('personal_certificate_label')) ID(user ID)
```

13.6 Hardware cryptography and Java cryptography providers

The *Java Cryptography Extension (JCE)* is a set of Java packages that provides a framework and implementations for encryption, key generation, key agreement, and *Message Authentication Code (MAC)* algorithms. Support for encryption includes symmetric, asymmetric, block, and stream ciphers. JCE is a pluggable framework that allows other providers (security packages) signed by a trusted entity to be plugged into the JCE framework, allowing new algorithms to be added seamlessly.

Vendors, such as SUN and IBM, provide their implementations of JCE. The IBM version of JCE (IBMJCE) is an implementation of the JCE cryptographic service provider that is compatible in z/OS environments. The IBMJCE is similar to SunJCE with a few differences. IBMJCE has a different package naming convention, it offers more algorithms, and it supports z/OS-specific keystores.

The IBMJCECCA is an extension to IBMJCE that adds the capability to use hardware cryptography through IBM *Common Cryptographic Architecture (CCA)* interfaces. IBMJCECCA provides secure, high-speed cryptographic services on various platforms through hardware cryptographic devices.

IBMJCECCA requires the following hardware and software:

- A system at the z/OS V1R6 level or later with either:
  - On a z800 or z900 processor, a Common Connector Framework (CCF) and a PCICC card
  - On a z890 or z990 processor, a CPACF and a PCIXCC card
  - On a z890 or z990 processor, a CPACF and a CEX2C card
  - On a z9® or z10 processor, a CPACF and a CEX2C or CEX2A card
- ICSF must be running.

There are a variety of other security providers that can be used with the JCE architecture. The focus of this section is on the IBMJCE and IBMJCECCA providers in a WebSphere 7.0 z/OS environment.

13.6.1 Choosing a JCE provider

A *JCE provider* is a Java package or set of Java packages that supplies a concrete implementation of a subset of cryptography aspects of the Java security API. The provider is typically packaged in a jar file and placed on the class path for use by the Java virtual machine (JVM) during startup. The providers that the JVM use are listed in a `java.security` file that is located in the HFS and are instantiated when the JVM starts. Beginning with WebSphere V6.0 for z/OS, the JVM is shipped as part of WebSphere. For WebSphere V7.0 for z/OS, the provider jars and `java.security` file are in these locations:

- 31-bit JVM:
  - `<WAS_HOME>/java/lib/security` (location of java.security file)
  - `<WAS_HOME>/java/lib` (location of provider jars)
- 64-bit JVM:
  - `<WAS_HOME>/java64/lib/security` (location of java.security file)
  - `<WAS_HOME>/java64/lib` (location of provider jars)

<WAS_HOME> is the location of the WebSphere configuration HFS.

Important: In a Network Deployment environment, the JVMs of the deployment manager and node agent run in 64-bit mode by default (since WebSphere V7.0). You can choose between 31-bit and 64-bit mode for each application server setup. As result, the environment is mixed in most cases. Therefore, we recommend that you change the `java.security` file in the Java 31-bit path and the 64-bit path.

Example 13-4 shows a list of providers in the `java.security` file. IBMJCECCA and IBMJCEFIPS are commented out in this sample. The providers are listed in order by preference, and the first provider IBMJCE is used as the default provider.

Example 13-4 Providers listed by preference

```
# List of providers and their preference orders (see above):
#
#security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.provider.IBMJCE
security.provider.2=com.ibm.jsse.IBMJSSEProvider
security.provider.3=com.ibm.jsse2.IBMJSSEProvider2
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
security.provider.5=com.ibm.security.cert.IBMCertPath
security.provider.6=com.ibm.security.sasl.IBMSASL
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
security.provider.8=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.9=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.10=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.11=org.apache.harmony.security.provider.PolicyProvide
r
```

A new default provider can be chosen by setting the provider name as the first provider in the list, enumerating the order of existing security providers and saving the file. WebSphere needs to be restarted to pick up any changes that have been made to the `java.security` file.

13.6.2 Administrative console keystore types

In previous versions of WebSphere, the keystore types available were only those programmed into the administrative console. In WebSphere V7.0, the administrative console dynamically builds its list of available keystore types based on the providers listed in the `java.security` file. The `java.security` file is read during server startup.

In Example 13-5, the `java.security` file was modified to include only the IBMJCE provider.

Example 13-5 Modified file

```
# List of providers and their preference orders (see above):
#
security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCE
```

The following list contains the keystore types that are available for the IBMJCE provider:

- JKS
- JCEKS
- JCERACFKS
- PKCS12JarSigner
- PKCS12
- CMSKS
- PKCS11

In Example 13-6, the `java.security` provider was modified to include an additional hardware cryptography provider IBMJCECCA.

Example 13-6 The java.security file

```
# List of providers and their preference orders (see above):
#
security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
security.provider.2=com.ibm.crypto.provider.IBMJCE
```

After the application server has been restarted, you can select the additional keystore types JCECCARACFKS and JCECCAKS when creating a new keystore. The keystore types JCECCARACFKS and JCECCAKS appear in the list as a result of adding the IBMJCECCA provider to the front of the list in the `java.security` file.

Figure 13-18 shows the additions to the keystore list. Select **SSL certificate and key management** → **Key stores and certificates** → **New**.

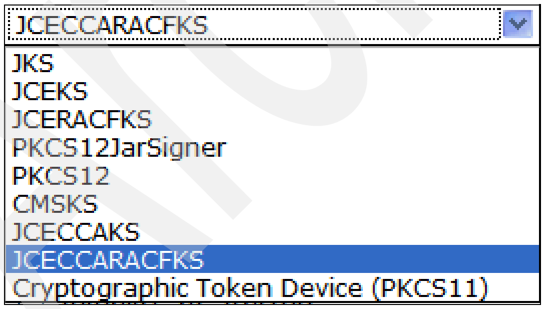


Figure 13-18 Keystore types from the IBMJCECCA and IBMJCE providers

When using the IBMJCECCA hardware cryptography provider, the unrestricted jar files need to be installed on the WebSphere system, and the ICSF address space must be up and running prior to starting WebSphere.

Keystores and truststores

The *keystore* is a database that can contain public and private keys. In WebSphere, the public keys are stored as signer certificates, while the private keys are stored in personal certificates. The keys are used for a variety of purposes, including authentication and data integrity.

A *truststore* is a type of keystore whose database contains trusted certificates that belong to another party. In WebSphere, a trusted certificate and a public key are stored as a signer certificate. The certificates are considered trusted, because the truststore owner trusts that the public key in the certificate belongs to the identity that has been identified by the subject of the certificate. The public key certificate can be used to confirm that a person is really who the person claims to be and that the data really came from that person. In general terms, truststores and keystores are often referred to as *keystores*.

13.6.3 IBMJCECCA and IBMJCE characteristics

In WebSphere on distributed systems, the certificates in a keystore are usually stored in a file. In WebSphere V7.0 for z/OS, the certificates in a keystore can be stored in a file or in the SAF database. The actual public/private keys for the corresponding certificates can be stored in a file, in a SAF database, or in hardware, depending on the provider and the keystore used. The IBMJCE and IBMJCECCA provider supports various keystore types that can be selected depending on the use of the location of certificates and keys.

Table 13-3 on page 524 describes the characteristics of the IBMJCE and IBMJCECCA providers based on the keystore types. The Use HW column indicates that hardware is used for SSL acceleration. We assume RACF as the SAF provider, but you can use any other security product accessible with SAF interfaces, such as Top Secret and Advanced Communications Function 2 (ACF2).

Table 13-3 JCE characteristics

Provider name	Keystore type	Certificate location	Key location	Uses HW	WebSphere path	Tooling
IBMJCECCA IBMJCE	JKS JCEKS PKCS12	File	File system	Yes	path/file	WebSphere ikeyman keytool
	JCECCARACFKS	RACF	Hardware	Yes	safkeyring:/// safkeyringhw:///	RACF
	JCECCAKS	File	Hardware	Yes	path/file	hwkeytool
IBMJCE	JKS JCEKS PKCS12	File	File system	No	path/file	WebSphere ikeyman keytool
	JCERACFKS	RACF	RACF	No	safkeyring:///	RACF

Note: To use hardware cryptography, both the IBMJCECCA and IBMJCE providers need to be listed in the `java.security` file with the IBMJCECCA provider listed first. Refer to “Update the java.security file with the IBMJCECCA provider” on page 529 for an example of this list.

Keystore descriptions

The following list contains a description of each of the keystore types from Table 13-3:

- Java KeyStore file (JKS)

  This traditional file-based keystore format is available with the standard JDK™. Files based on this format are usually stored with a `.jks` extension.

  Use this option if the keystore file is stored in JKS format.

- Java Cryptography Extension Keystore (JCEKS)

  JCEKS is a file-based keystore implementation of the Java Cryptography Extensions provider. Files based on this format are usually stored with a `*.jceks` extension.

  Use this option if the keystore file is stored in JCEKS format.

- Public Key Cryptography Standards version 12 Keystore (PKCS12KS)

  This file format is commonly used to store private keys with accompanying public key certificates protected with a password-based symmetric key. Files based on this format are usually stored with a `.p12` extension.

  Use this option if your keystore uses the PKCS#12 file format.

- JCE RACF Keystore (JCERACFKS)

  This keystore is used for certificates that are stored in a SAF keyring and the keys are stored in RACF.
- JCE Common Cryptographic Architecture RACF Keystore (JCECCARACFKS)

  This keystore is used for certificates that are stored in a SAF keyring, and the keys can be stored in RACF or stored in hardware and are ICSF-managed.
- JCE Common Cryptographic Architecture Keystore (JCECCAKS)

  This file-based keystore is used when the certificates are stored in a file, but the keys are in hardware that is managed with the hwkeytool utility.
- Public Key Cryptography Standards version 11 Keystore (PKCS11KS)

To better understand how to use certificates that are stored in RACF, refer to 13.6.4, “SSL and JCERACFKS keystore” on page 525 and 13.6.5, “Hardware cryptography using a JCECCARACFKS keystore” on page 526. These sections provide two examples that lead you through setting up and accessing certificates in RACF with a safkeyring URI.

13.6.4 SSL and JCERACFKS keystore

This example demonstrates creating a signer certificate and a personal certificate, attaching them to a keyring, and configuring the administrative console to use that keyring. This example assumes that the provider is set to the default IBMJCE and that the keys are managed in software.

These actions can be performed with the following RACF commands:

1. Generate a signer certificate labeled WebSphereCA:

   ```
   RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('WebSphere CA') OU('IBM'))
   WITHLABEL('WebSphereCA')
   ```
2. Create a personal certificate signed by WebSphereCA:

   ```
   RACDCERT ID (WPACR) GENCERT SUBJECTSDN(CN('wtsc04.itso.ibm.com')
   O('IBM')) WITHLABEL('WPCell Application Server CR')
   SIGNWITH(CERTAUTH LABEL('WebSphereCA')) TRUST
   ```
3. Create a ring called WPKeyring for the CR’s user ID:

   ```
   RACDCERT ADDRING(WPKeyring) ID(WPACR)
   ```
4. Connect the WebSphereCA to the ring WPKeyring:

   ```
   RACDCERT ID(WPACR) CONNECT (RING(WPKeyring) LABEL('WebSphereCA')
   CERTAUTH)
   ```

5. Connect the personal certificate to WPKeyring:

   ```
   RACDCERT ID(WPACR) CONNECT (LABEL('WPCell Application Server CR')
   RING(WPKeyring) DEFAULT)
   ```

13.6.5 Hardware cryptography using a JCECCARACFKS keystore

This example demonstrates creating a signer certificate and a personal certificate with the keys in hardware and the setup needed in the administrative console to use these certificates. This example assumes that the default provider is set to IBMJCECCA and that ICSF is currently running.

This exercise includes these steps:

1. Create the certificates in RACF with keys in the hardware.
2. Install unrestricted policy jars in WebSphere.
3. Update the `java.policy` file with the IBMJCECCA provider.
4. Create a keystore in the administrative console.

Keyring and certificate setup with keys in hardware

To set this up:

1. Create a new signer certificate called `WPHDCA` with its key in hardware:

   ```
   RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('wtsc04.itso.ibm.com')
   OU('IBM')) WITHLABEL('WPHDCA') TRUST ICSF
   ```

2. Create a new keyring called `WPKeyring.HD` for the control user ID `WPACR` and servant user ID `WPASR`:

   ```
   RACDCERT ADDRING(WPKeyring.HD) ID(WPACR)
   RACDCERT ADDRING(WPKeyring.HD) ID(WPASR)
   ```

3. Connect the signer certificate `WPHDCA` to `WPKeyring.HD` for `WPACR` and `WPASR`:

   ```
   RACDCERT ID(WPACR) CONNECT (RING(WPKeyring.HD) LABEL('WPHDCA')
   CERTAUTH)

   RACDCERT ID(WPASR) CONNECT (RING(WPKeyring.HD) LABEL('WPHDCA')
   CERTAUTH)
   ```

4. Create a new personal certificate `HDPersonal` signed by `WPHDCA` with its key in hardware:

   ```
   RACDCERT ID (WPACR) GENCERT SUBJECTSDN(CN('HDPersonal') O('IBM'))
   WITHLABEL('HDPersonal') SIGNWITH(CERTAUTH LABEL('WPHDCA')) TRUST
   ICSF
   ```

5. Connect the new personal certificate to the CR's keyring:

   ```
   RACDCERT ID(WPACR) CONNECT (LABEL('HDPersonal') RING(WPKeyring.HD)
   DEFAULT)
   ```

The commands in step 3 connect the WPHDCA signer certificate to both the CR and the SR keyrings. It might not be necessary to connect the signer certificate to the SR keyring, depending on the endpoint for SSL communication. We perform this step so that the signer certificate can be viewed in the administrative console.

The certificates are appended with the ICSF keyword, which indicates that RACF needs to attempt to store the private key associated with this certificate in the ICSF PKDS.

Example 13-7 shows a display of the signer certificate information:

```
RACDCERT CERTAUTH LIST(LABEL('WPHDCA'))
```

Example 13-7 Signer certificate WPHDCA information

```
Digital certificate information for CERTAUTH:

  Label: WPHDCA
  Certificate ID: 2QiJmZmDhZmjgebXyMTDwUBA
  Status: TRUST
  Start Date: 2009/21/02 00:00:00
  End Date:   2012/12/31 23:59:59
  Serial Number:
       >00<
Issuer's Name:
       >CN=HDwtsc04.itso.ibm.com.OU=IBM<
  Subject's Name:
       >CN=HDwtsc04.itso.ibm.com.OU=IBM<
  Key Usage: CERTSIGN
  Private Key Type: ICSF
  Private Key Size: 1024
  PKDS Label: IRR.DIGTCERT.CERTIFAUTH.SC04.C3D1494CDB090586
  Ring Associations:
    Ring Owner: WPACR
    Ring:
       >WPKeyring.HD<
    Ring Owner: WPASR
    Ring:
       >WPKeyring.HD<
```

Example 13-8 shows a display of the personal certificate information:

```
RACDCERT LIST (label('HDPersonal')) ID(WPACR)
```

Example 13-8 HDPersonal certificate information

```
Digital certificate information for user WPACR:

  Label: HDPersonal
  Certificate ID: 2QXm18HD2cjE14WZopaVgZNA
  Status: TRUST
  Start Date: 2009/21/02 00:00:00
  End Date:   2012/30/12 23:59:59
  Serial Number:
       >01<
Issuer's Name:
       >CN=wtsc04.itso.ibm.com.OU=IBM<
  Subject's Name:
       >CN=HDPersonal.O=IBM<
  Private Key Type: ICSF
  Private Key Size: 1024
  PKDS Label: IRR.DIGTCERT.WPACR.SC04.C3D1494D6FDD2480
  Ring Associations:
    Ring Owner: WPACR
    Ring:
       >WPKeyring.HD<
```

Installing the unrestricted Java policy jars

The WebSphere JVM ships with policy jars that provide only limited function cryptography. Certain countries have restrictions on the import, re-export, possession, and use of encryption software. To use the IBMJCECCA provider with hardware, the unrestricted policy jars that provide full function cryptography need to be downloaded and installed in WebSphere.

To obtain the Java unrestricted policy jars and place them on the WebSphere z/OS system:

1. Obtain the unrestricted policy jars from this URL:

   `http://www.ibm.com/developerworks/java/jdk/security/index.html`

2. Make a backup of the original restricted `local_policy.jar` and `US_export_policy.jar` files.
3. Download the unrestricted `local_policy.jar` and `US_export_policy.jar` files to your WebSphere z/OS system in `<WAS_HOME>/AppServer/java64/lib/security`.

4. After copying the `local_policy.jar` and `US_export_policy.jar` files, change the permissions so that the CR and the SR address spaces can access the jar files:

   – `chmod 644 local_policy.jar`
   – `chmod 644 US_export_policy.jar`

   Note: In WebSphere V7.0, the 64-bit support is enabled by default for the deployment manager in a Network Deployment environment. Therefore, you have to choose the correct Java directory in the <*WAS_HOME*> path.

Update the java.security file with the IBMJCECCA provider

The `java.security` file that is located in the WebSphere HFS needs to be updated with the IBMJCECCA provider as the first provider in the list of providers. The `java.security` file is in ASCII, so it might be necessary to convert the file to EBCDIC first before making changes.

Example 13-9 illustrates the addition made to the `java.security` file.

Example 13-9 Updating the java.security file for IBMJCECCA

```
# List of providers and their preference orders (see above):
#
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.sasl.IBMSASL
security.provider.8=com.ibm.security.cmskeystore.CMSProvider
security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.10=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.11=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.12=org.apache.harmony.security.provider.PolicyProvide
r
```

In a Network Deployment environment, the `java.security` file must be adjusted in every node.

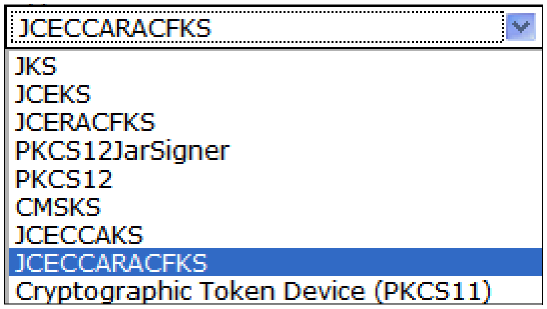


Figure 13-19 The JCECCARACFKS provider will be displayed

Administrative console setup

In the WebSphere for z/OS administrative console, create a new keystore by selecting **SSL certificate and key management** → **Key stores and certificates** → **New**. Enter this information:

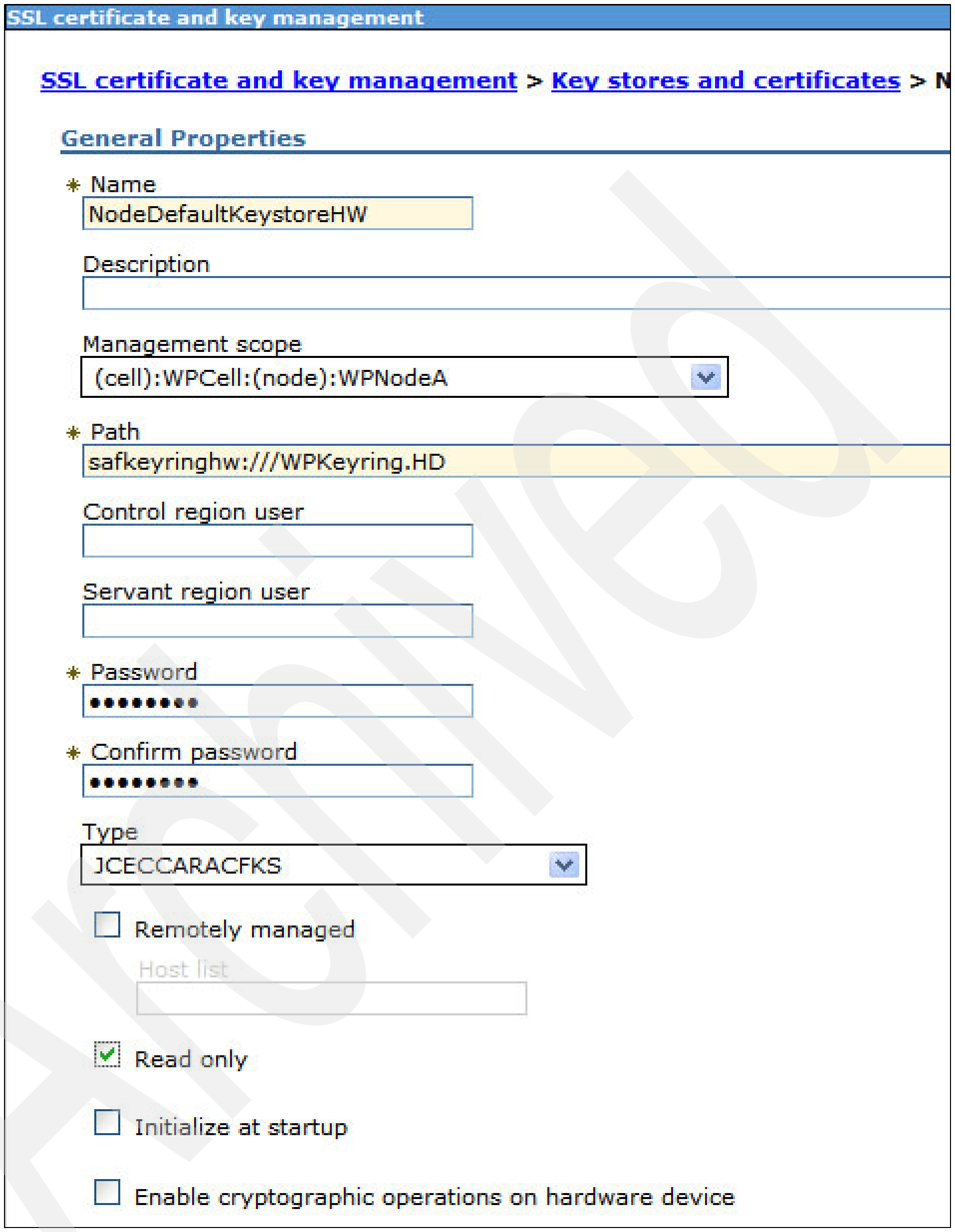
- Name: `JCECCARACFKS KeyStore`
- Path: **`safkeyringhw:`**`///WPKeyring.HD`
- Password: `password`
- Confirm password: `password`
- Type: **JCECCARACFKS**
- Check **Read only**.

Then, click **Apply**, and save the changes.

The safkeyring URI needs to be specified as `safkeyringhw` to indicate that the keys for the certificates in the keyring are stored in hardware. Avoid any trailing spaces after the `safkeyringhw` URI. The password for accessing certificates in RACF is `password`.

When setting up the certificates, the WPHDCA certificate was connected to both the CR and SR user ID so that the certificate information can be viewed in the administrative console.

Figure 13-20 on page 531 shows the options that were chosen in the administrative console when setting up the keystore.



SSL certificate and key management

SSL certificate and key management > Key stores and certificates > N

General Properties

* Name
NodeDefaultKeystoreHW

Description

Management scope
(cell):WPCell:(node):WPNodeA

* Path
safkeyringhw:///WPKeyring.HD

Control region user

Servant region user

* Password
••••••••

* Confirm password
••••••••

Type
JCECCARACFKS

☐ Remotely managed

Host list

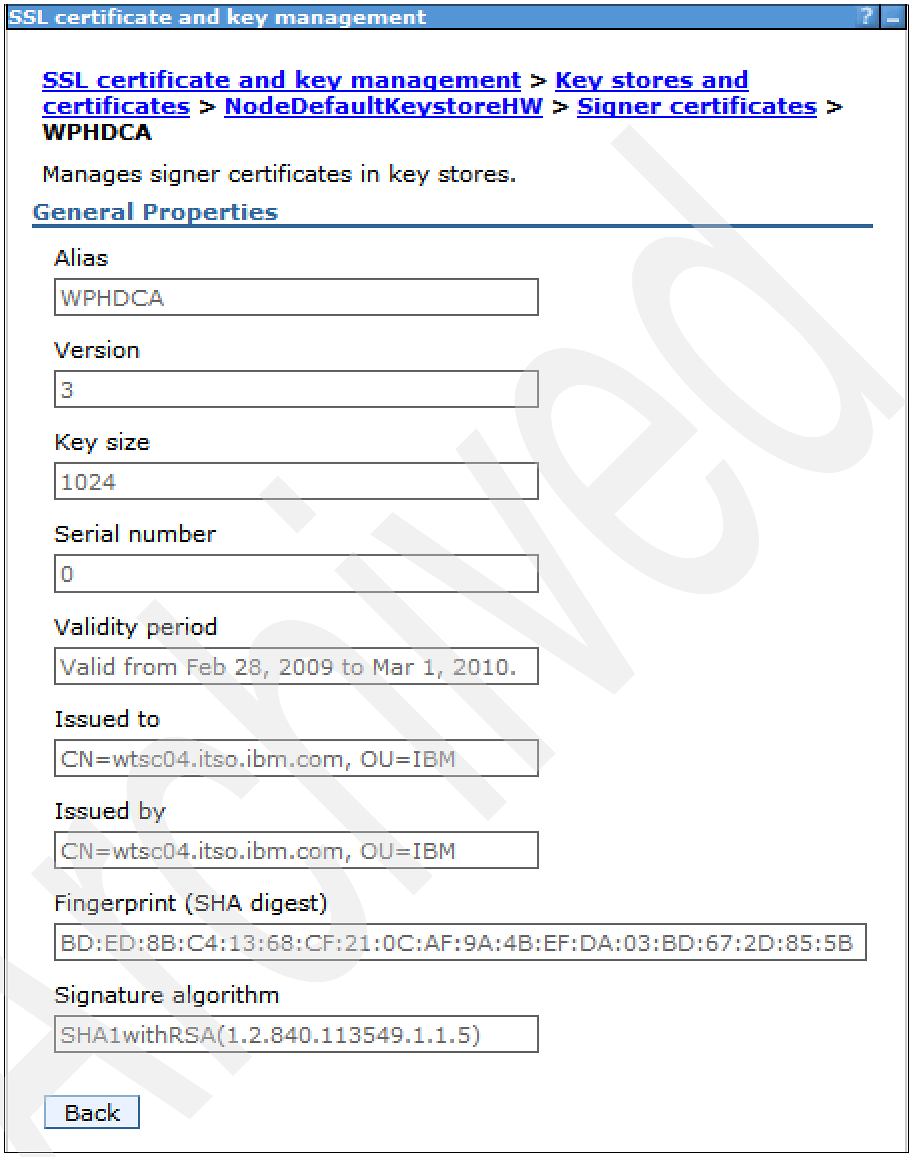
☑ Read only

☐ Initialize at startup

☐ Enable cryptographic operations on hardware device

Figure 13-20 Creating a JSSE configuration using a safkeyring URI

Figure 13-21 shows the signer certificate information for keyring `WPKeyring.HD`.



SSL certificate and key management > Key stores and certificates > NodeDefaultKeystoreHW > Signer certificates > **WPHDCA**

Manages signer certificates in key stores.

General Properties

Alias	WPHDCA
Version	3
Key size	1024
Serial number	0
Validity period	Valid from Feb 28, 2009 to Mar 1, 2010.
Issued to	CN=wtsc04.itso.ibm.com, OU=IBM
Issued by	CN=wtsc04.itso.ibm.com, OU=IBM
Fingerprint (SHA digest)	BD:ED:8B:C4:13:68:CF:21:0C:AF:9A:4B:EF:DA:03:BD:67:2D:85:5B
Signature algorithm	SHA1withRSA(1.2.840.113549.1.1.5)

Figure 13-21 Signer certificate WPHDCA information

More information: You can obtain additional information about hardware cryptographic configurations, including performance benchmarks for WebSphere Application Server for z/OS, in the white paper *SSL Options in WebSphere for z/OS V6.1*, WP101213. Moreover, this white paper provides an additional jython script, which defines keystore, truststore, and JSEE configuration using the IBMJCECCA provider. A second jython script assigns the defined JSEE configuration to an HTTPS port of an application server. The white paper is available at the following Web site:

`http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101213`

13.7 SSL troubleshooting and traces

SSL handshake errors are one of the most common problems that can surface when attempting to set up secure communications in WebSphere. This section provides diagnostic steps that can be performed to identify an incorrect SSL setup, the types of traces to gather to diagnose them, and common problems.

13.7.1 Diagnostic steps

When attempting to diagnose SSL handshake issues, follow these steps to identify the cause of the problem:

1. Identify the endpoints for the SSL communication. It is important to determine who the SSL client (client initiating the SSL handshake) is and who the SSL server (the receiving party in the SSL handshake attempt) is. Identifying the SSL client and the SSL server is sometimes difficult, because the SSL client can be a distributed Java 2 Platform, Enterprise Edition (J2EE) client or thin client, a Web browser, a WebSphere distributed process, or a z/OS deployment manager or application server CR or SR address space. The SSL server is usually WebSphere on a distributed process or WebSphere on a z/OS CR.
2. Determine the SSL configuration for the SSL client and the SSL server. SSL handshake issues occur between two endpoints, and the SSL handshake error usually surfaces in the job output or error log of the client.

   If the SSL client or server is on z/OS, obtain the user ID of the address space or process. A keyring that contains the certificates is associated with the user ID. Knowledge of the certificates attached to those keyrings can be used to identify where the setup might be incorrect.

If the SSL client or server is on a distributed network, it is necessary to obtain the SSL configuration (keystore, truststore, and certificates).

A list of example scenarios is provided to demonstrate where to obtain the SSL configuration for client and server:

- SSL handshake issue that occurs during synchronization between the deployment manager and node agent:
  - Deployment manager CR user ID
  - Node agent CR user ID
- SSL handshake issue attempting to connect to the administrative console:
  - SSL configuration of off-platform Web browser
  - Deployment manager CR user ID (Network Deployment) or application server CR user ID (base configuration)
- SSL handshake issues when attempting to connect to a Web application:
  - SSL configuration of off-platform Web browser
  - Application server CR user ID
- SSL handshake issues when using HFS clients (that is, wsadmin.sh, syncNode.sh, launchClient.sh, and so on):
  - User ID executing the client on z/OS
  - User ID of administrative console on which it is running (either the user ID of the deployment manager CR (Network Deployment) or the user ID of the application CR user ID (base))

3. For a z/OS client or server, display the certificates on the keyring associated with the user ID and provide the certificate details for the signer certificate and personal certificate (if there is one).

   For the distributed client or server, display the certificate information for the signer certificate and personal certificate (if there is one) using the administrative console SSL management, or iKeyman utility.

4. From the displayed information, check the validity period for the signer certificate on the SSL client side and SSL server side. Verify that the current date falls within the start date and end date for the certificates. The certificate's start date must precede the current date and not be expired.

5. From the displayed information, confirm that the SSL client's truststore contains the signer certificate of the server. Verify that the issuer's distinguished name and subject distinguished name for the server's signer certificate on the SSL client side match those of the server personal certificate on the SSL server side.

13.7.2 SSL traces

When SSL errors surface, and analyzing the certificates, keyrings, and WebSphere setup does not provide enough information to diagnose the problem, SSL traces can be enabled in either the WebSphere administrative console or in the JVM.

WebSphere SSL trace

When diagnosing WebSphere SSL problems, SSL traces can be enabled in WebSphere and in the JVM.

Follow the path in the administrative console in WebSphere: **Logging and Tracing** → ***server_name*** → **Change Log Detail Levels**. Click the **Configuration** tab and add the WebSphere trace specification:

```
SSL=all
```

The WebSphere trace can also be turned on dynamically from the MVS console by using the following modifying command:

```
F CR_short_name,TRACEJAVA='SSL=all'
```

Reset the trace to the original trace specification at startup:

```
F CR_short_name,TRACEINIT
```

Java JSSE trace

To enable diagnostic trace for determining the cause of SSL handshake errors, follow the path in the administrative console to **Application servers** → ***server_name*** → **Process Definition** → **Control/Servant** → **Java Virtual Machine**.

In the Generic JVM Arguments field, add:

```
-Djavax.net.debug=true -Djava.security.auth.debug=all
```

The Java trace setting cannot be enabled dynamically and requires the application server to be restarted to pick up the change.

13.7.3 Common errors

The following section includes common errors that can occur when attempting SSL handshake communication. The examples provide the cause of the problem that resulted in the error message:

- Error message when a signer certificate is missing from a client's truststore in order to attempt an SSL handshake. This type of error surfaces on the client initiating the SSL handshake:

```
CWPKI0022E: SSL HANDSHAKE FAILURE:  A signer with SubjectDN
"CN=wtsc04.itso.ibm.com, OU=SC04, O=IBM" was sent from target
host:port "*:9044".  The signer may need to be added to local trust
store
"C:/Program1/IBM/WebSphere/AppServer/profiles/AppSrv01/config/cells/
Node02Cell/nodes/Node02/trust.p12" located in SSL configuration
alias "NodeDefaultSSLSettings" loaded from SSL configuration file
"security.xml".  The extended error message from the SSL handshake
exception is: "No trusted certificate found".
```

- The following error text shows a problem that can occur using the IBMJCECCA provider with safkeyring URI. The ICSF address space has become unavailable while a user is attempting to access the administrative console or application:

```
Message: BBOO0220E: SSLC0008E: Unable to initialize SSL connection.
Unauthorized access was denied or security settings have expired.
Exception is javax.net.ssl.SSLHandshakeException: Client requested
protocol Unknown 0.2 not enabled or not supported.
com.ibm.ws.ssl.channel.impl.SSLHandshakeErrorTracker
```

- The following error text shows a problem while starting WebSphere using the IBMJCECCA provider, but with the ICSF address space not running:

```
java.lang.RuntimeException: Hardware error from call CSNBOWH
returnCode 12 reasonCode 0
com.ibm.crypto.hdwrCCA.provider.SHA.a(Unknown Source)
com.ibm.crypto.hdwrCCA.provider.SHA.engineDigest(Unknown Source)
java.security.MessageDigest$Delegate.engineDigest(MessageDigest.java
:554)
java.security.MessageDigest.digest(MessageDigest.java:332)
com.ibm.ws.management.repository.DocumentDigestImpl.calc(DocumentDig
estImpljava:128)
com.ibm.ws.management.repository.FileDocument.calcDigest(FileDocumen
t.java:212)
```

14

Security identity propagation

This chapter discusses the use of security identity propagation, the Sync-to-Thread Allowed option, and propagation of the client user IDs to CICS and DB2.

The following topics are discussed here:

14.1 Sync-to-Thread Allowed and RunAs thread identity

The *Sync-to-Thread* security concept is unique to WebSphere Application Server on z/OS. Application Sync-to-Thread provides the capability for an application to synchronize the RunAs identity being used, with the identity on the native z/OS thread. This identity is used by SAF to access resources (such as hierarchical file system (HFS) files) outside of the application server that are controlled by the operating system. This capability is provided for EJBs and for Web applications.

In more detail, the servant region's task control block (TCB)-level accessor environment element (ACEE) is set to the current Java Platform, Enterprise Edition (Java EE) thread identity. This synchronization is effective as long as the EJB or Web application is running the current request. When the EJB or Web completes processing, the native thread is restored to its former state.

The Application Sync-to-Thread feature is only supported for local operating system (localOS) registries and for Lightweight Directory Access Protocol (LDAP) registries with a corresponding System Authorization Facility (SAF) mapping module. The following conditions must all be true:

- The configured user registry is the local operating system (LocalOS).
- The Sync-to-Thread allowed value is set to `true`.

  To set this value using the administrative console, go to **Security** → **Global security** → **z/OS security options**.

  A global switch is also provided for each server so that the ability to synchronize the identities can be enabled or disabled at the server level.

  If synchronization is disabled, the server identity is always used.
- The application must be configured to run with application Sync-to-Thread. This setting is included in the application deployment descriptor as an env-entry of `com.ibm.websphere.security.SyncToOSThread={true|false}`. When this entry is not defined in the deployment descriptor, the default value is `false`.
- The RACF administrator must define a FACILITY class profile and optional SURROGAT class profile to ensure that Sync-to-Thread Allowed is utilized.

14.1.1 Required RACF profiles for Sync-to-Thread

In order to enable Sync-to-Thread, additional RACF profiles have to be created. If the application server has been created with the z/OS product option, the generated customization job contains the command to define the FACILITY class profile, but it does not contain the permit commands to that profile, nor the define and permit commands for the SURROGAT class profiles.

The control region (CR) user ID needs READ or CONTROL access to enable Sync-to-Thread. With READ access, only security environments representing users in the SURROGAT class are allowed, while CONTROL allows for security environments to represent any RACF-defined user ID.

Create the Sync-to-Thread profile according to your naming conventions:

```
RDEFINE FACILITY BBO.SYNC.<cell_short_name or
saf_profil_prefix>.<cluster_short_name> UACC(NONE)

PERMIT BBO.SYNC.<cell_short_name or
saf_profile_prefix>.<cluster_short_name> CLASS(FACILITY) ID(UID_CR)
ACC(READ or CONTROL)
```

If the control region has only READ permissions to BBO.SYNC, an additional SURROGAT class profile is required for implementing Sync-to-Thread:

```
RDEFINE SURROGAT BBO.SYNC.<UID_SR> UACC(NONE)
PERMIT BBO.SYNC.<UID_SR> CLASS(SURROGAT) ID(J2EE identity) ACC(READ)
```

The Java principal (Java EE identity) requesting the synchronization of the OS thread needs READ access to a surrogate profile. In this case, UID_SR stands for the user ID under which the application server's servant region (SR) is running. Again, this SURROGAT profile is optional. It will only be checked if the control region user ID has READ access to the BBO.SYNC profile.

The following article from the WebSphere Information Center provides a table of all supported resource adapters in combination with Sync-to-Thread:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/cdat_conthid.html

14.1.2 Sync-to-Thread example

In this example, a simple test application attempts to access a UNIX file named /tmp/test.txt. This test application is secured by the basic authentication method, and the application user ID USER3 is permitted to the required Java EE role.

For testing purposes, we set the UNIX file permissions for /tmp/test.txt so that both the SR user ID WPASR and application user ID USER3 do *not* have access.

Let us see what our security violation messages will render when Synch to OS thread is disabled in the first example and is enabled in the second example. In both cases, `USER3` logs on to the test application with its credentials.

When running with Sync-to-Thread disabled, the following security violation occurs:

```
ICH408I USER(WPASR) GROUP(WPCFG ) NAME(WAS APPSVR SR       )
/tmp/test.txt CL(FSOBJ ) ID(01E6E2C8C6E2F800010300003C3C0000)
INSUFFICIENT AUTHORITY TO OPEN ACCESS INTENT(R--) ACCESS ALLOWED(OTHER
---)  EFFECTIVE UID(0000002113)  EFFECTIVE GID(0000002300)
```

In this case, the started task control (STC) User `WPASR` attempted to access file `/tmp/test.txt`.

When running with Sync-to-Thread enabled, the following security violation occurs:

```
ICH408I USER(USER3   ) GROUP(GROUP1  ) NAME(CB390 USER3          )
/tmp/test.txt CL(FSOBJ ) FID(01E6E2C8C6E2F800010300003C3C0000)
INSUFFICIENT AUTHORITY TO OPEN ACCESS INTENT(R--) ACCESS ALLOWED(OTHER
---)   EFFECTIVE UID(0000033114)  EFFECTIVE GID(0000033333)
```

This security violation shows that propagated user ID `USER3` attempted to access file `/tmp/test.txt`.

14.1.3 SAF delegation

SAF delegation minimizes the need to store user IDs and passwords in many locations in the configuration. SAF delegation is used in conjunction with the RunAs feature.

WebSphere Application Server for z/OS V7.0 supports delegation, allowing a user identity to be represented as a Java EE role. For example, you can establish an application to be run with a RunAs role of `RoleA`. `RoleA` can then be mapped as `UserA`. WebSphere then establishes the principal as `UserA`, and `RoleA` is defined in the deployment descriptor. With this configuration, SAF delegation uses the specified Java EE role `RoleA` to determine the user identity and then sets the Java principal with the user ID, `UserA`. `UserA` is specified in the SAF EJBROLE's profile APPLDATA value of the RDEFINE RACF command. The RDEFINE command in this example looks like this command:

```
RDEFINE EJBROLE rolea UACC(NONE) APPLDATA(usera)
```

SAF delegation requires that SAF authorization is enabled. The SAF security administrator is responsible for the assignment of users to the role.

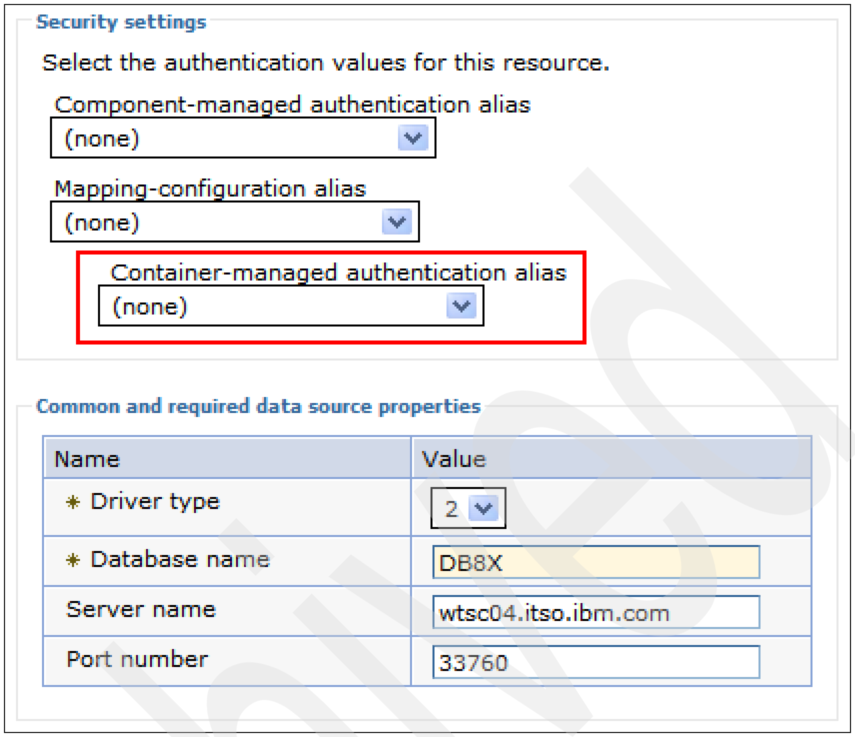
This option is available in the administrative console by selecting **Security** → **Global security** → **External authorization provider**, selecting **SAF authorization options**, and clicking **Configure**.

14.2 Propagating user credentials to DB2 using JDBC Type 2 driver

WebSphere Application Server V7.0 for z/OS allows you to assign a thread identifier as an owner of a connection. After the Connection Manager performs the synchronization, the native z/OS thread identity is temporarily replaced with the Java thread identity. The Java thread identity is the identity that is used to obtain the Enterprise Information System (EIS) connection. The Connection Manager Sync-to-Thread support provides a way to obtain an EIS connection using the RunAs identity. After obtaining the connection, the Connection Manager restores the previous OS thread identity.

In order to create the thread identity function for the Java Database Connectivity (JDBC) provider data sources, follow these steps:

1. For user ID propagation to DB2, the `BBO.SYNC` profile is required according to the RACF definitions in “Required RACF profiles for Sync-to-Thread” on page 538.
2. Assure that the **DB2 Universal JDBC Driver Provider** is used and that the JDBC Type 2 connection to the local DB2 database works properly.
3. As shown in Figure 14-1 on page 542, set the Container-managed authentication alias to **none** on the data source. If a container-managed authentication alias is specified at this point, this user ID and password will be used instead of the propagated user ID.


Security settings

Select the authentication values for this resource.

Component-managed authentication alias: (none)

Mapping-configuration alias: (none)

Container-managed authentication alias: (none)

Common and required data source properties

Name	Value
* Driver type	2
* Database name	DB8X
Server name	wtsc04.itso.ibm.com
Port number	33760

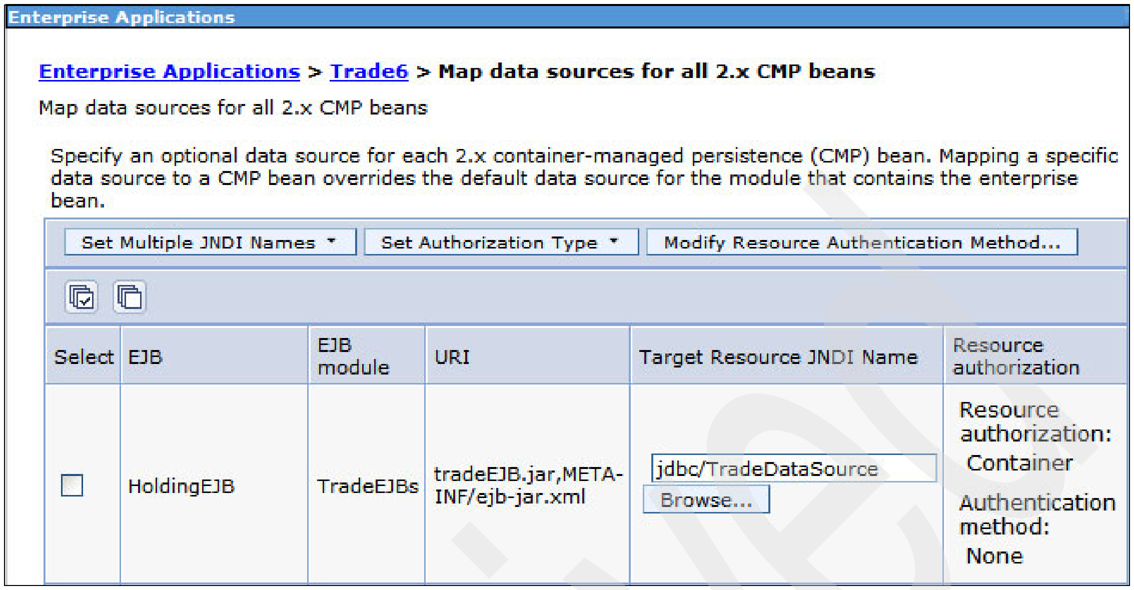
Figure 14-1 Data source settings

4. All data source resource references within the application must be set to:

   – Resource authorization: **Container**
   – Authentication method: **none**

   These settings can be changed in the deployment descriptor of the application.

   Alternatively, the settings can be changed after application deployment in the administrative console. Select **Enterprise Applications** → ***application_name*** as shown in Figure 14-2 on page 543. A resource reference can be changed to res-auth: Container by selecting the resource reference, clicking **Set Authorization Type**, and finally, selecting the authorization type **Container**. Moreover, the authentication method can be set to **none** by clicking **Modify Resource Authentication Method**. This procedure needs to be done for each data source resource reference in that application.


Enterprise Applications

Enterprise Applications > Trade6 > **Map data sources for all 2.x CMP beans**

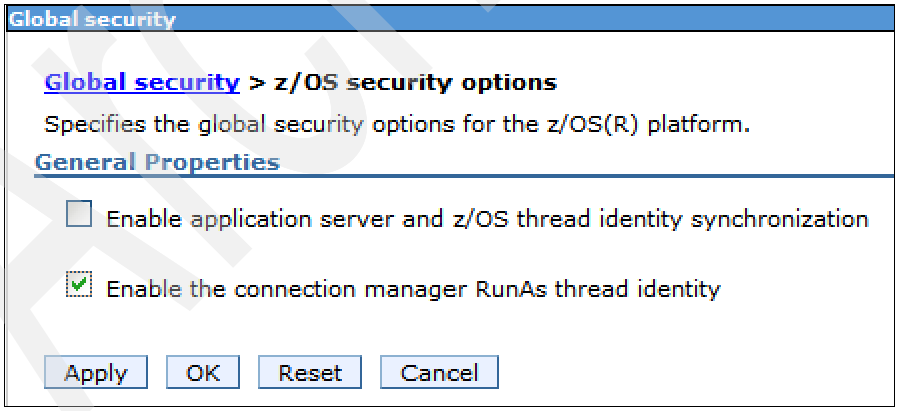
Map data sources for all 2.x CMP beans

Specify an optional data source for each 2.x container-managed persistence (CMP) bean. Mapping a specific data source to a CMP bean overrides the default data source for the module that contains the enterprise bean.

Select	EJB	EJB module	URI	Target Resource JNDI Name	Resource authorization
☐	HoldingEJB	TradeEJBs	tradeEJB.jar,META-INF/ejb-jar.xml	jdbc/TradeDataSource Browse...	Resource authorization: Container Authentication method: None

Figure 14-2 Change Res-auth Container in the administrative console

5. Enable the connection manager RunAs thread identity by using the following path in the administrative console: **Security** → **Global security** → **z/OS security options**. Click **Enable the connection manager RunAs thread identity.** This check box is the major switch to enable security propagation to DB2 and additionally can be defined on a server-by-server basis.


Global security

Global security > **z/OS security options**

Specifies the global security options for the z/OS(R) platform.

General Properties

☐ Enable application server and z/OS thread identity synchronization

✓ Enable the connection manager RunAs thread identity

Figure 14-3 Global switch for enable RunAS thread identity for DB2

6. To enable identity propagation, an authentication method must be enabled and the Java EE roles must be defined in the deployment descriptor of the application. In addition, the corresponding EJBROLE profiles need to be created in RACF, and the application users need to be mapped to these EJBROLE profiles. If authentication for the application is disabled in the deployment descriptor, the unauthenticated user (guest user) will be used to access DB2.

   In Example 14-1, we modified the deployment descriptor of the trade application using Rational Application Developer Assemble and Deploy and then redeployed the application.

Example 14-1 Extract from the deployment descriptor of the trade application

```
<security-constraint>
   <display-name>trade</display-name>
      <web-resource-collection>
         <web-resource-name>tradeResourceCollection</web-resource-name>
         <url-pattern>/*</url-pattern>
         <http-method>GET</http-method>
         <http-method>POST</http-method>
      </web-resource-collection>
      <auth-constraint>
         <description>UACC to trade application</description>
         <role-name>TradeUACC</role-name>
      </auth-constraint>
</security-constraint>
<login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>Trade Login</realm-name>
</login-config>
<security-role>
   <role-name>TradeUACC</role-name>
</security-role>
```

   According to the Java EE role in the deployment descriptor, a new EJBROLE `WP.TradeUACC` is created in RACF using the following command:

```
RDEFINE EJBROLE WP.TradeUACC UACC(NONE)
```

   A new RACF group `TRADEGRP` is used to assign the users to the role:

```
AG TRADEGRP OWNER(SYS1) SUPGROUP(SYS1) DATA('TRADER')
```

   Users of that application are assigned to the defined role by getting connected to this group. Assure that these users have sufficient permissions to access the DB2 tables of the application:

```
CONNECT JVESER GROUP(TRADEGRP)
```

Finally, the new RACF group is granted the `TradeUACC` role:

```
PERMIT WP.TradeUACC CLASS(EJBROLE) ID(TRADEGRP) ACCESS(READ)
```

7. Restart the cell so that the global security changes can take effect.
8. In order to verify the user ID propagation to DB2, invoke the application using the correct context root. Consequently, the application asks for a user name and password. This user ID will be used to access DB2. If the EJBROLE is set up correctly, the user will be able to log in. Invoke several database queries in the application.
9. The following MVS console command is an example of displaying all current active DB2 threads with the appropriate authentication user ID. Use this command to verify the user ID propagation to DB2:

   ```
   D8X1 DIS THD(*)
   ```

   The output of this command displays all active DB2 threads, each in its own row. The column `ID` identifies the started task name of the request origin and the `AUTHID` is the provided user ID. In this case, the `AUTHID` matches the user ID from the application login.

Example 14-2 Output for DB2 thread command

```
DSNV402I  -D8X1 ACTIVE THREADS - 346
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    TD    1938 WPS02AS      JVESER   ?RRSAF   010E 151954
RRSAF    N        1              LDAPSRV           0081      0
RRSAF    T        5 LDAPD8XG     LDAPSRV  DSNACLI  0081 60674
```

14.3 Propagating credentials to CICS

Thread identity support allows WebSphere Application Server for z/OS to automatically pass the user ID of the Java thread to CICS Transaction Server when using the External Call Interface (ECI) resource adapter. The setting of the thread user ID is dependent on the RunAs policy for the J2EE application. If RunAs is set to Caller, and the user of the Web application has authenticated with the WebSphere Application Server, the thread identity support enables the caller's identity to be propagated to CICS automatically. Thread identity support enables WebSphere Application Server to behave in a way that traditional z/OS address spaces behave. After the user ID has been authenticated, that user ID flows with any work done within the z/OS system, which is similar to the way that multiregion operation (MRO) requests in CICS are managed, for example.

In this topology, the ECI resource adapter sends only the user ID to CICS Transaction Server with the ECI request. It is assumed that the user is already authenticated by WebSphere Application Server. A trust relationship can be configured between WebSphere Application Server and the CICS server using the MRO bind-and-link security mechanisms. Surrogate security checks can also be enabled to confirm that the user ID associated with the WebSphere Application Server region has the appropriate authority to flow a specific user ID (or one of a generic set of user IDs) to CICS Transaction Server. Figure 14-4 on page 548 shows this architecture.

In contrast to user ID propagation to DB2, there is no global switch to turn on security propagation for CICS.

The following conditions must exist to enable user ID propagation to CICS:

- CICS Transaction Gateway is installed. A local connection is used between WebSphere Application Server and CICS.
- Global security is enabled, and the active user registry is defined as Local OS.

14.3.1 Application-related settings

The following actions must be taken for the application:

- Application security has to be enabled in WebSphere.
- A security constraint that contains the Java EE roles must be defined in the application deployment descriptor. The corresponding EJBROLEs must be defined in RACF. If authentication for the application is disabled in the deployment descriptor, the unauthenticated user (guest user) will be used to access CICS.
- Set res-auth to Container in the EJB deployment descriptor.

  In the deployment descriptor of the EJB (`ejb-jar.xml`), the res-auth value of the ECI resource references must be set to `Container` (Example 14-3).

  Example 14-3 Res-auth setting in the EJB deployment descriptor

  ```
  <resource-ref id="ResourceRef_1">
     <description>CICS ECI Resource adapter</description>
     <res-ref-name>ECI</res-ref-name>
     <res-type>javax.resource.cci.ConnectionFactory</res-type>
     <res-auth>Container</res-auth>
  </resource-ref>
  ```

- Set RunAS to *Caller* in the EJB deployment descriptor.

A RunAs setting on a servlet or EJB affects the Java thread identity of methods that are invoked on a subsequent call. WebSphere Application Server RunAs policy allows three choices in assigning the Java thread identity for the current request:

– *Caller* uses the caller's identity for the method selected and to propagate it to any subsequent methods invoked or J2EE resources accessed. Caller is the default behavior.

– *Server* indicates that the method will assume the identity of the WebSphere SR.

– *Role* means that the application assembler selects the name of a security role that is defined in the application. Authorization is performed by checking that the principal name has been assigned to one of the required security roles.

14.3.2 Required RACF profiles

Create the RACF profiles:

- Assure that the SR has access to the CICS region.

  In the FACILITY class, the DFHAPPL profile restricts the access to the CICS region. If multiple application servers need access to CICS, the SR group can be permitted instead. In this profile, the *<APPLID>* is the APPLID of the CICS region:

```
PERMIT DFHAPPL.<APPLID> CLASS(FACILITY) ID(GID_SR) ACCESS(READ)
SETRPTS RACLIST(FACILITY) REFRESH
```

- Assure that the application users have access to the corresponding CICS transaction. In addition, the application server SR needs to be permitted to the CICS transaction as well. This example grants the access to a CICS transaction:

```
PERMIT MIRROR CL(TCICSTRN) ID(UID_SR) ACCESS (READ)
PERMIT MIRROR CL(TCICSTRN) ID(UID) ACCESS (READ)
SETROPTS RACLIST(TCICSTRN) REFRESH
```

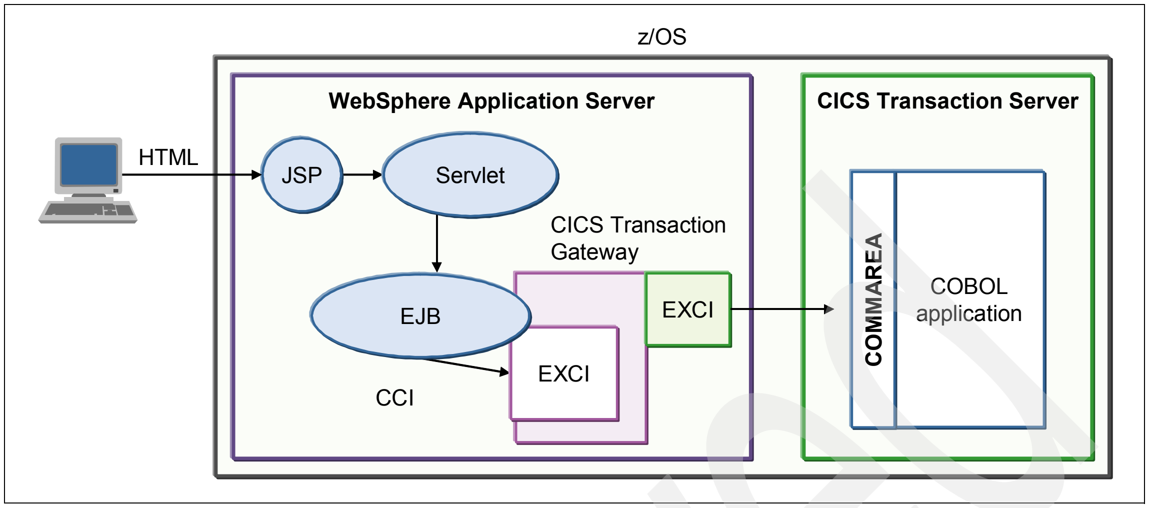


Figure 14-4   WebSphere and the CICS Transaction Gateway deployed on z/OS

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

For information about ordering these publications, refer to "How to get IBM Redbooks publications" on page 551. Note that several of the documents referenced here might be available in softcopy only:

- *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316
- *WebSphere Application Server V7 Web Services Guide*, SG24-7758
- *WebSphere Application Server V7 Messaging Administration Guide*, SG24-7770
- *Understanding LDAP - Design and Implementation*, SG24-4986
- *Distributed Security and High Availability with Tivoli Access Manager and WebSphere Application Server for z/OS*, SG24-6760

Online resources

These Web sites are also relevant as further information sources:

- WebSphere Application Server V7 Information Center

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp`
- WebSphere Application Server V6.1 Information Center

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp`
- IBM Tivoli Information Center

  `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp`
- WebSphere Application Server V6 advanced security hardening: Part 1

  `http://www.ibm.com/developerworks/websphere/techjournal/0512_botzum/0512_botzum1.html`

- WebSphere Application Server V6 advanced security hardening: Part 2

  http://www.ibm.com/developerworks/websphere/techjournal/0512_botzum/0512_botzum2.html
- The full J2EE Connector architecture can be downloaded from Sun Microsystems at:

  http://java.sun.com/j2ee/connector/
- Choosing among JCA, JMS and Web services for EAI

  http://www.ibm.com/developerworks/webservices/library/ws-jcajms.html
- IBM Education Assistant for WebSphere Application Server V6 Security: CSIv2

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v6/was/6.0/Security/WASv6_Sec_CSIv2/player.html
- Object Management Group (OMG) Security Web site, CSIv2 topic

  http://www.omg.org/technology/documents/formal/omg_security.htm#CSIv2
- The Common Object Request Broker: Architecture and Specification

  http://www.omg.org/docs/formal/98-12-01.pdf
- Common Annotations for the Java Platform
  http://jcp.org/aboutJava/communityprocess/final/jsr250/index.html
- Enterprise JavaBeansTM,Version 3.0 EJB Core Contracts and Requirements
  http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html
- IBM developerWorks Security information Web pages

  http://www.ibm.com/developerworks/java/jdk/security/60/
  http://www.ibm.com/developerworks/java/jdk/security/index.html
- MustGather: Errors using iKeyman with IBM HTTP Server

  http://www-01.ibm.com/support/docview.wss?uid=swg21202820
- System Requirements for WebSphere Application Server V7.0

  http://www-01.ibm.com/support/docview.wss?uid=swg27006921
- List of supported software for WebSphere Application Server V7.0

  http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27012369
- *White paper WP10653 - WebSphere z/OS -- WSC Sample ND Configuration*:

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100653
- WebSphere for z/OS Version 7 - Configuration Planning Spreadsheets

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3341

- Configuring Fine Grained Security in WebSphere Application Server V6.1 for z/OS

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD10332
  4

- z/OS Security Server RACF Callable Services

  http://publibz.boulder.ibm.com/epubs/pdf/ichzd180.pdf

- SSL Options in WebSphere for z/OS V6.1

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP10121
  3

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM

Redbooks

WebSphere Application Server V7.0 Security Guide

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

WebSphere Application Server V7.0 Security Guide

Secure WebSphere administration processes

Ensure secure WebSphere applications

Secure communication with SSL

This IBM Redbooks publication provides the information that is needed to implement secure solutions with WebSphere Application Server V7.0. It focuses on security for the application server and its components, including enterprise applications.

This book includes administrative and infrastructure security, application security, and z/OS specifics.

This book is intended for anyone who plans to secure applications and the application serving environment.

SG24-7660-00 ISBN 0738433004