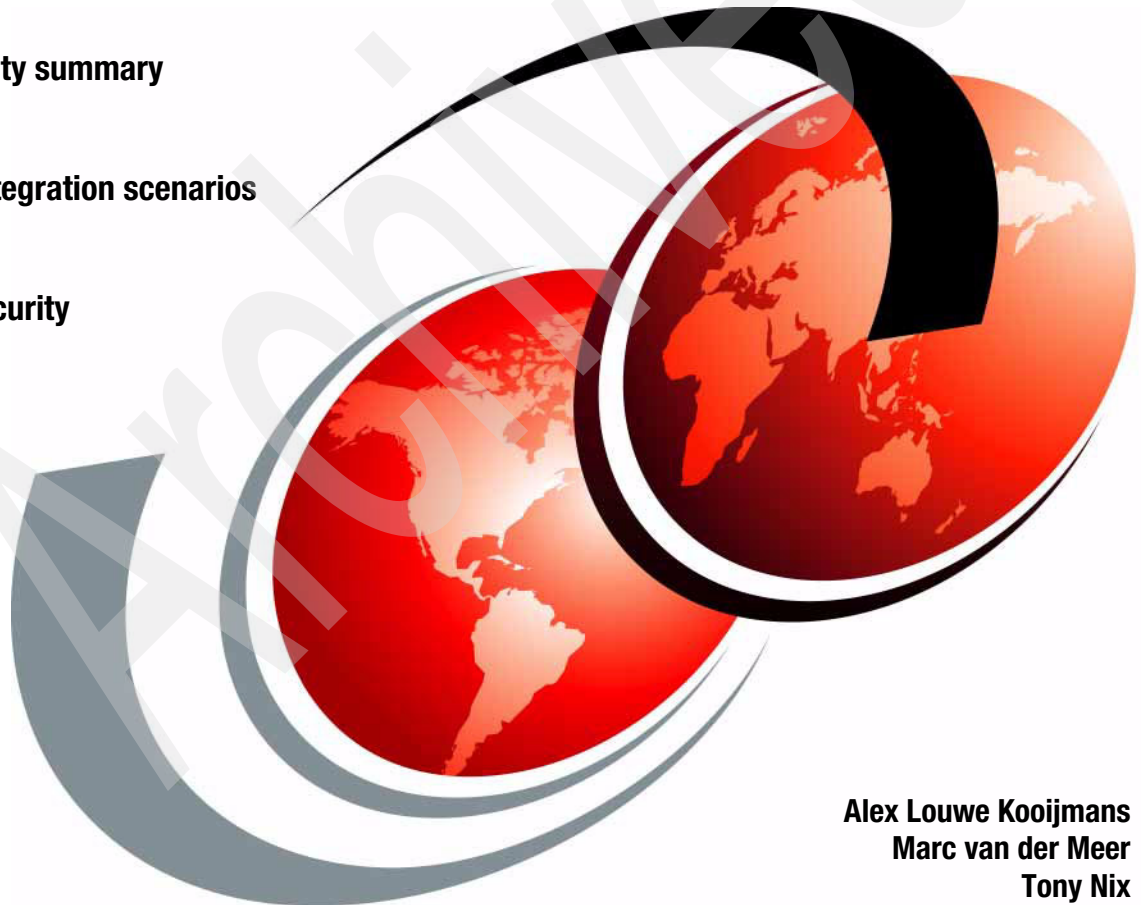


# WebSphere Application Server on z/OS and Security Integration

z/OS security summary

Security integration scenarios

Bridged security scenario



Alex Louwe Kooijmans  
Marc van der Meer  
Tony Nix





International Technical Support Organization

**WebSphere Application Server for z/OS and  
Security Integration**

June 2006

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

## **First Edition (June 2006)**

This edition applies to WebSphere Application Server for z/OS Version 6.02.

© Copyright International Business Machines Corporation 2006. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team that wrote this Redpaper .....	ix
Become a published author .....	xi
Comments welcome .....	xi
<b>Chapter 1. Introduction</b> .....	1
1.1 Audience and objective of this Redpaper .....	2
1.2 Security challenges .....	3
1.3 Impact of security challenges on mainframe-centric environments .....	5
1.3.1 Application topology .....	5
1.3.2 Starting points for a secure design .....	7
1.4 How to come to an integrated solution .....	8
1.4.1 Administration .....	8
1.4.2 Authentication .....	9
1.4.3 Authorization .....	10
1.4.4 Auditability .....	10
1.4.5 Propagation of security credentials .....	11
1.5 Conclusions .....	11
<b>Chapter 2. End-to-end security scenarios</b> .....	13
2.1 Simple HTTP with application server .....	14
2.2 Extended HTTP with application server and LDAP SDBM backend .....	15
2.3 External authentication and LDAP native authentication .....	16
2.4 External authentication, external authorization, and LDAP native authentication .....	17
2.5 End-to-end, authentication to authorization chain .....	18
2.6 J2EE client .....	19
2.7 Bridged security between z/OS and distributed .....	20
2.8 MQ client .....	21
<b>Chapter 3. z/OS and WebSphere security technology overview</b> .....	25
3.1 General concepts .....	26
3.1.1 Single sign-on .....	26
3.1.2 Credential .....	26
3.1.3 Principal .....	26
3.1.4 Security domain .....	27

3.1.5 Subject .....	27
3.2 The evolution of security integration .....	27
3.2.1 Classic view .....	28
3.2.2 The introduction of WebSphere Application Server .....	30
3.2.3 Further hardening through a reverse proxy .....	33
3.2.4 Beyond HTTP and HTTPS .....	36
3.2.5 Advanced security concerns .....	38
3.3 What are the (new) security points of interest? .....	40
3.4 Related security technology .....	41
3.4.1 LTPA .....	42
3.4.2 REXX executables to add RACF definitions .....	42
3.4.3 User registry .....	42
3.4.4 Global security .....	43
3.5 LDAP on z/OS .....	44
3.5.1 Distinguished names .....	44
3.5.2 Native authentication .....	44
3.5.3 User registry .....	44
3.6 Administration .....	45
3.6.1 Administrative console .....	45
3.6.2 z/OS Security Server (RACF) .....	45
3.6.3 LDAP .....	46
3.6.4 Naming conventions .....	46
3.6.5 Practices .....	47
3.6.6 Procedures .....	47
3.6.7 Architectural suggestions .....	47
3.7 Authentication .....	47
3.7.1 User IDs .....	47
3.7.2 Passwords .....	48
3.7.3 Reauthentication .....	48
3.8 Authorization .....	49
3.8.1 Role-based security design .....	49
3.8.2 RACF resources .....	50
3.9 Auditability .....	50
3.9.1 Corporate requirements .....	50
3.9.2 Carrying forward security credentials .....	51
<b>Chapter 4. A sample solution .....</b>	<b>53</b>
4.1 LDAP SDBM registry .....	54
4.1.1 Setting up LDAP with an SDBM backend .....	54
4.1.2 WebSphere user registry settings .....	55
4.1.3 Security flow .....	56
4.1.4 Benefits of an LDAP SDBM solution .....	57
4.2 Security integration between z/OS and distributed .....	58

4.2.1 Bridged security solution . . . . .	59
4.2.2 Overview of the benefits of a bridged security solution . . . . .	69
<b>Appendix A. J2EE security</b> . . . . .	71
J2EE 1.4 security features . . . . .	72
Java 2 security . . . . .	72
Java Authentication and Authorization Service (JAAS) . . . . .	76
J2EE security roles . . . . .	77
Declarative security . . . . .	78
Deployment descriptor . . . . .	78
JAAS (Java Authentication and Authorization Service) . . . . .	78
Permission-based model . . . . .	78
Programmatic security . . . . .	78
Roles . . . . .	79
RunAs . . . . .	79
CSlv2 . . . . .	80
Java Authorization Contract for Containers (JACC) . . . . .	82
Putting together J2EE security . . . . .	84
<b>Appendix B. z/OS Security Server (RACF)</b> . . . . .	87
Technology . . . . .	88
Components . . . . .	88
z/OS Security Server (RACF) . . . . .	89
Cryptographic Services . . . . .	89
Integrated Security Services . . . . .	90
Usability . . . . .	92
Adherence to industry standards . . . . .	92
ACEE . . . . .	92
RACO (ENVR, Environment Object) . . . . .	93
Groups . . . . .	93
Resource classes . . . . .	93
Auditing . . . . .	94
System Authorization Facility (SAF) . . . . .	94
User IDs . . . . .	95
Certificate Name Filtering . . . . .	95
RACF classes . . . . .	96
Back-end systems (EIS) . . . . .	98
DB2 . . . . .	98
<b>Related publications</b> . . . . .	101
IBM Redbooks . . . . .	101
Other publications . . . . .	101
Online resources . . . . .	102
How to get IBM Redbooks . . . . .	102

Help from IBM .....	102
<b>Index</b> .....	103

Archived



# Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:  
*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

# Trademarks

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

AIX®  
CICS®  
DB2®  
Domino®  
IBM®  
IMS™

MVS™  
RACF®  
Redbooks™  
Redbooks (logo) ™  
SecureWay®  
System z™

Tivoli®  
VTAM®  
WebSphere®  
z/OS®  
zSeries®

The following terms are trademarks of other companies:

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

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

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

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

# Preface

This IBM® Redpaper addresses the need for information in the area of integrating security between WebSphere® Application Server on z/OS® and the outside world.

In most cases, multiple security registries exist within a company with a different scheme of identities. This is even more likely in companies using z/OS. There are basically two “worlds”: the z/OS (RACF®) world in which identities and their authorizations are kept in RACF and the outside world where identities and their authorizations are kept in LDAP, Microsoft® Active Directory®, or equivalent solutions.

In an e-business environment, the first authentication of a user is usually already performed before a request reaches the z/OS environment based on an ID not known in that exact form on z/OS. There are basically two challenges, and both of them are addressed in this paper:

- ▶ Authenticate a user on a distributed server and be able to trust that user when coming into WebSphere Application Server on z/OS.
- ▶ Propagate the user ID and eventual security credentials from the distributed environment to WebSphere Application Server on z/OS, and eventually transform the ID and credentials to something that is administered and understood on z/OS.

## The team that wrote this Redpaper

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

**Alex Louwe Kooijmans** is a project leader at the International Technical Support Organization, Poughkeepsie Center. He leads residencies to create Redbooks™ in the area of Java™ and WebSphere Application Server on z/OS. He also instructs workshops in this area. Alex is on assignment to the ITSO for the second time. Before joining the ITSO, he worked in various other roles, such as IT Specialist supporting customers in Europe getting started with WebSphere Application Server on various platforms, and Client IT Architect in the Financial Services Sector in the Netherlands. He has a lot of experience in Java and WebSphere Application Server on z/OS, and integrating WebSphere Application

Server with DB2®, MQ, CICS®, and IMS™ in particular. He has worked for IBM since 1986.

**Marc van der Meer** is an IT specialist working for Global Technology Services in the Netherlands. His work domain includes general z/OS and subsystems and has eight years of experience. He has been involved in several security migrations and has specialized in RACF security. He is working on WebSphere Application Server on z/OS with one of the larger mainframe clients in the Netherlands in the role of architect, debugger of connectors and security.

**Tony Nix** is a Certified Senior IT Specialist currently working with the zBlue Software Migration Project Office (SMPO) team in Costa Mesa, California. He has 26 years of IT experience in a variety of areas, including computer operations, systems and applications programming, project management, line management, security administration, auditing, training, and consulting. As a member of the SMPO for 10 years, Tony has been involved in many diverse mainframe security migrations. He holds an external CISSP certification (Certified Information Systems Security Professional) and internal IBM IT Specialist certification.

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

Richard Conway  
IBM International Technical Support Organization, Poughkeepsie Center

Foulques de Valence  
IBM Global Services - France, Northwest Africa Infrastructure Solutions -  
Business Flexibility

Thomas Koman  
IBM WebSphere Application Server Information Development

Peggy LaBelle  
IBM z/OS Security (EIM and RACF) Development and Test

Colette Manoni  
IBM WebSphere z/OS Development

Thomas Muenz  
IBM SWG Software Services WebSphere

Gary Puchkoff  
WebSphere Application Server for z/OS Design and Development

Carl Wohlers  
WebSphere for z/OS Specialist

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll 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](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an e-mail to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- ▶ Mail your comments to:

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



# Introduction

This chapter sets the scene before we start discussing more technical details. Starting with a discussion of the paper's audience and objective, we follow with a description of the challenges we face in designing a good solution for security and the impact of those challenges on a mainframe-centric environment.

We recommended design points for the mainframe-centric security solution, and conclude with a wrap-up of the main points.

## 1.1 Audience and objective of this Redpaper

The audience for this Redpaper includes users who have a well-established culture around z/OS and are looking to deploy new applications on a WebSphere Application Server for z/OS Version 6 or later.

**Attention:** Most information presented in this paper also applies to WebSphere Application Server V5. However, we only validated the information against WebSphere Application Server for z/OS Version 6.

The following individuals will primarily benefit from the content of this Redpaper:

- ▶ Software architects
- ▶ Security system designers
- ▶ System z™-centric (z/OS) IT professionals
- ▶ Security administrators and analysts curious about how WebSphere could be implemented on z/OS
- ▶ Professionals with distributed system knowledge who want more understanding of how WebSphere could be deployed in a z/OS environment

When applications are first developed, there is usually a lot of focus on what the new application will do, and sometimes not a lot of thought is put into how the application is integrated for production use within the enterprise. This paper offers information about the decisions that are involved when configuring and subsequently deploying security for a WebSphere Application Server. The primary focus of this paper is to give the audience guidance about how to integrate WebSphere Application Server into a z/OS culture by providing examples of scenarios that can be found in a distributed WebSphere Application Server culture. We also discuss authentication, authorization, and auditability of users, groups, subjects, principals, and processes. We emphasize the advantages of deploying on z/OS for those customers who are already “zCentric,” including working in a strictly z/OS topology and in a heterogeneous topology.

This Redpaper describes different scenarios in multiple layers to give enough detail about what is truly involved with each scenario. This is not intended to be a “how-to” document, but it provides architectural guidance to ensure a consistent security deployment of WebSphere applications running on z/OS.

The scope of this paper is to describe a high-level architectural approach for designing a “reasonable” level of security using WebSphere Application Server for z/OS Version 6, z/OS Security Server (RACF), LDAP, and related back-end subsystems such as CICS, IMS, DB2, and MQSeries.



This paper is organized as follows:

- ▶ In this chapter we outline the real challenges we are facing today in a mainframe-oriented environment and show that existing security technology on the mainframe combined with new security technology outside the mainframe can be a very good fit to address those challenges.
- ▶ In Chapter 2, “End-to-end security scenarios” on page 13 we present the most common security integration architectures with WebSphere Application Server on z/OS.
- ▶ Chapter 3, “z/OS and WebSphere security technology overview” on page 25 provides a summary of the security technologies that are available to implement security integration solutions.
- ▶ Chapter 4, “A sample solution” on page 53 explains in more detail one of the most appealing solutions to access WebSphere Application Server on z/OS securely.

## 1.2 Security challenges

Security in a mainframe-centric environment in the past has been straightforward, where users would log on to the system and their applications with a 3270 terminal, using an RACF user ID and password. Each RACF user ID has a profile in which authorizations and privileges are kept. Because most of the applications have typically been in CICS or IMS, a fine-grained security model is available for IMS and CICS with which you can control access to the system or an application, as well as to a certain transaction or program.

So, in those days the “only” thing required was an RACF administrator maintaining access to the system, its applications, and its resources. To make things easy, RACF users are grouped under RACF groups.

In today’s world, in most enterprise solutions, the technology described above is no longer not enough because of the following transitions:

- ▶ Applications have become more complex, and the logical tiers in the applications are implemented in different servers and on different platforms. Hence, there is more communication within an application, which in many cases goes beyond the boundaries of the mainframe platform.
- ▶ The user interface is by preference implemented in different channels, such as a browser, a cell phone, or voice. Those devices are per definition not on the mainframe platform, but need to be able to integrate with the mainframe because the real business logic and data remains on the mainframe for obvious reasons.

- ▶ Security artifacts have to be accessible from multiple places and in many cases from multiple servers and platforms. This calls for security registry solutions that are standardized and accessible both locally and remotely.
- ▶ Government and corporate rules have been tightened significantly over the past few years, resulting in:
  - More and better auditability requirements
  - Strict access management to a company's assets and information
- ▶ With the escalating use of the Internet, more and more “self-service” applications have been introduced, where the customer is the user of the application and not simply an employee of the enterprise. The perfect example for this is Internet banking, where the bank application is accessible from any browser on the Internet, instead of only from a bank terminal or desktop operated by a bank employee.

This situation has had an enormous impact on today's security infrastructure, as all kinds of untrusted access points have been introduced to the infrastructure that have access to the enterprise's sensitive program logic and data on the mainframe.

As you might imagine, end-to-end security architecture is very important and essential before going any farther with the deployment of an application. When designing a new IT solution for a business problem, many security-related questions have to be answered, such as:

- ▶ Are the access points to the application's infrastructure trusted or untrusted?
- ▶ Who are the users who potentially can get access to the infrastructure or application? Can the user be anybody on the Internet, regular customers, or only employees?
- ▶ With which authentication mechanism do the users come into the environment?

Some authentication mechanisms are weaker than others. For example, user ID and password is considered to be a weak authentication mechanism, and a hard token is considered to be strong.

- ▶ Is there a requirement for single sign-on (SSO)?

Single sign-on is a solution in which a user authenticates just once for all applications the user is authorized to use during a certain time period.

- ▶ Does the original user ID and eventual credentials need to be propagated throughout the application?

In some cases, the original user ID and even credentials might be required farther along in the application or when accessing the data finally. Propagation could be required for:

- Auditing purposes
- Re-authentication in another tier of the application
- Authorization for certain functions of the application in any of the tiers

The answers to these questions usually trigger a discussion about which solution to choose. In the next section we discuss the impact of the just-discussed challenges on mainframe-centric environments.

## **1.3 Impact of security challenges on mainframe-centric environments**

In the following sections we discuss how the challenges mentioned in 1.2, “Security challenges” on page 3 affect the mainframe environment and how new applications should be designed from a security perspective.

### **1.3.1 Application topology**

Before we translate the challenges to an existing mainframe environment, we show an example of the most common application topology in Figure 1-1 on page 6, using a browser for the user interface, an HTTP server and application server in the middle, and a Transaction Monitor and Database Systems in the backend.

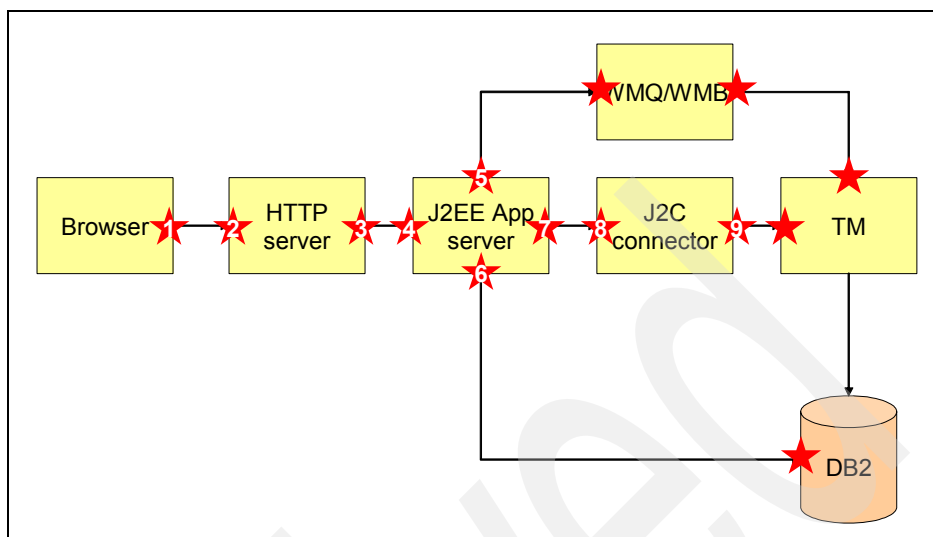


Figure 1-1 Example of the most common application topology

In this figure:

- ▶ Each star represents a *security interaction point*, where things can be configured to influence the security behavior. For example, this can be encryption, the invocation of an authentication user exit, or simply the propagation of a user ID or credential.
- ▶ Each tier can be on any platform, but we will discover that where you put each tier makes a difference from a security point of view.
- ▶ We did not mention any security components yet, such as firewalls, proxy servers, security registries, and so on; we position those later on in this paper.

The components depicted in Figure 1-1 are required in most modern business applications:

#### **Browser**

The standard user interface environment in today's IT world. The browser can also be located in a mobile device, such as a smartphone.

#### **HTTP server**

If there is a browser, there is also an HTTP server. The HTTP server manages the communication with the browser.

#### **J2EE™ application server**

The J2EE application server is not a hard requirement, but it is needed if there is a requirement to implement any of the application logic in J2EE or use portal or process functionality.

<b>J2C connector</b>	A J2C connector is required if the integration between the J2EE application server and the back-end system needs to be synchronous and transactional.
<b>WMQ/WMB</b>	WebSphere MQ or WebSphere Message Broker are solutions for message-based integration between the J2EE application server and the back-end system.
<b>TM</b>	Most business applications on the z/OS side of the mainframe contain a considerable portion of business logic implemented in CICS or IMS.
<b>DB2</b>	DB2 is the most common database used on the mainframe.

### 1.3.2 Starting points for a secure design

For any new business application on the mainframe or the Web-enablement of an existing application, the security impact has to be analyzed. Keeping the application topology in mind as discussed in 1.3.1, “Application topology” on page 5, new points of interest are introduced. So, what should be our starting points for this discussion? The following considerations are valid in most situations:

- ▶ We probably would not want to change the security model implemented in the back-end systems, just for the purpose of Web-enablement or for technical reasons. The way the access to back-end systems is currently arranged must to be taken into account when expanding or modernizing an application.
- ▶ Business applications of which the business logic will run on z/OS already have a z/OS Security Server oriented model of security. This means that RACF profiles already exist for certain resources and users of that application. The goal is to achieve a level of integration between this model and whatever outside model is being used.
- ▶ If new components are introduced and are placed outside the z/OS environment, they might create a potential threat and have to be secured. This will require additional security infrastructure, including firewalls, network configuration, and so on.
- ▶ A user seldom wants to authenticate multiple times before entering the application, so when additional tiers are introduced to the application this must not lead to multiple authentication actions. After first authentication, the user should be let into authorized functions without being asked again and again for authentication artifacts<sup>1</sup>.

---

<sup>1</sup> In some cases an application might require a stronger authentication mechanism than the one used in a previous authentication. In this case, the user must to authenticate again with this stronger mechanism, We call this “step up authentication.”

- Last, but not least, a new application landscape might introduce multiple places to administer security artifacts or, at least, multiple places to store those artifacts. Ideally, there is only one place and procedure to administrate the security artifacts and synchronize or replicate to other places if needed.

In most large enterprises running z/OS, besides z/OS Security Server, there will most likely also be Lightweight Directory Access Protocol (LDAP) and MicroSoft Active Directory based security registries. When applications have to be integrated, there will also be a need for security integration.

- With so many government and corporate rules requiring the ability to trace back transactions, auditability has become a very important aspect. Auditability means that transactions can be traced back to the original requestor. For example, a certain database update must have a user ID associated to it. This user ID may be a functional or group ID, but then other proof for the originating individual must exist, such as inside the data portion. In an application with multiple tiers (again, refer to “Application topology” on page 5 for a good example) this means that the identity of the original requester must be propagated to the end of the application chain.

## 1.4 How to come to an integrated solution

A good way to look at the new security solution will be to address the four “A’s”: *administration*, *authentication*, *authorization*, and *auditability*. For this discussion, we do not include privacy requirements.

### 1.4.1 Administration

Administration means having procedures and technology to keep security artifacts current and safe. Ideally, those artifacts only have to be changed, added, or deleted in one place. With the z/OS Security Server (RACF), z/OS provides a bulletproof environment for performing the administration of user IDs and other resources. However, the non-z/OS world is used to work with other types of registries, such as Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory.

The purpose of RACF, LDAP, and Active Directory is the same, but there are many functional differences between them.

Before being able to finally answer the question of which registry should be the *master*, it is important to know where the information in this registry is going to be used for enforcement of security. If authentication is performed outside z/OS on a Windows® or AIX® server and RACF is the master registry, then the information in RACF should be available on that Windows or AIX server. This can be

achieved through replication or synchronization techniques or in some cases through direct access to RACF on z/OS.

When using an LDAP-based authentication on any platform (Windows, AIX, and so on) it is in principle possible to perform this authentication against RACF information. (Refer to Chapter 4, “A sample solution” on page 53.)

Table 1-1 provides an overview of the most common combinations of master security directory and directory used for enforcement. The master registry is the directory where the security artifacts are kept primarily. “Enforcement” shows how and on which platform the application gains access to security artifacts.

*Table 1-1 Integration between security directories*

Master registry	Enforcement	Solution
z/OS Security Server	z/OS Security Server access	Standard z/OS security
z/OS Security Server	LDAP access on z/OS	Standard z/OS security
z/OS Security Server	LDAP access on distributed	Direct access to z/OS. Directory replication or synchronization between RACF and any LDAP
z/OS LDAP	z/OS Security Server	Standard z/OS security
z/OS LDAP	LDAP access on distributed	Standard LDAP
Distributed LDAP	z/OS Security Server access	Directory replication
Distributed LDAP	LDAP access on z/OS	Standard LDAP

## 1.4.2 Authentication

It is very likely that before a user comes into the z/OS environment, an authentication has already taken place somewhere. If the authentication is strong enough, we could say that there is *trust* and z/OS would not have to re-authenticate that same user.

After the user has authenticated, the individual process that the user has initiated (for example, a batch job, transaction, or JavaBean) is associated with the user preassigned identity. This association will follow the process logically within the operating system, and it is available to identify the authenticated user during access control authorization checking and for auditing purposes.

Digital certificates and Kerberos are two examples of trusted third-party identification and authentication techniques that are being used on z/OS. The Network Authentication service in z/OS is based on the Kerberos V5 protocol.

Public key infrastructure (PKI) provides a trusted infrastructure that can be managed and supports the use of digital certificates. PKI services are provided as part of the latest releases of z/OS. You can act as your own *certificate authority (CA)*.

Single sign-on might be required in some environments. Using IBM Tivoli® Access Manager can help to authenticate to different applications on different platforms without having to re-authenticate to each application again and again.

In most modern scenarios, a logical choice is to place an authentication proxy with SSO capabilities in front of the applications.

### 1.4.3 Authorization

Authentication only verifies the identity of a user or request, but it is not the same as authorization to use certain assets. A user might have been authenticated nicely, but might not have been allowed access to anything anywhere. After successful authentication to the z/OS environment, access should be provided only to the assets a user has permission to access. Authorization mechanisms are available in all tiers of an application and z/OS is very strong in enforcing authorization (access control). On z/OS you can provide authorization at a program or file level, and at a much more fine-grained level. For some resources on z/OS, access is not granted for any user, and are accessible only by authorized components of the operating system.

In an end-to-end application, authorization mechanisms can be used anywhere from the first Web page until the final update in a DB2 table on z/OS. Each technology used in each tier provides specific authorization techniques.

### 1.4.4 Auditability

One of the most important features of centralized authentication and authorization is the ability to record and analyze security information from a single focal point. This information is essential for ensuring that your security policy is being followed.

Laws and instructions regarding auditing and logging information are increasing. RACF has the capability to deal with those requirements. Because it can identify and verify a user's ID and recognize which resources the user can access, RACF can record events where user-resource interaction has been attempted. This function records actual access activities or variances from the expected use of the system.



### 1.4.5 Propagation of security credentials

In an integrated environment with strong auditability and authorization requirements, it is obvious that the user's original security credentials might have to remain available throughout the entire application, from the first HTTP send from the browser until the final update or insert into a database. However, the exact requirements for this vary and are dependent on the business requirements for the application and corporate security policies.

Even though the authentication takes place outside the z/OS platform and even though certain application tiers are not placed on z/OS, there are still many ways to take the security credentials forward from the distributed components onto the z/OS part of the application. There is no reason to offload application components from z/OS because they cannot be integrated with components on distributed systems. When a request arrives on z/OS, for example at the IBM HTTP Server, the credentials can be propagated further to the application server or even the back-end application in CICS or IMS. As long as the credentials are available, additional authentication (or reauthentication) and authorization actions can be performed. The application requirements determine at which point the original credentials can be dropped.

**Note:** In many environments, back-end systems are accessed without security context or with generic user IDs, and the original security context can be dropped in an early stage.

## 1.5 Conclusions

Within the z/OS environment, security is taken care of very well from an authentication, authorization, and auditability point of view. Administration can be performed with z/OS Security Server (RACF) or LDAP.

Nowadays, in most cases, an application request does not originate in the z/OS environment, but somewhere outside the environment; this puts additional integration requirements on the solution. A variety of solutions is available to integrate the bulletproof security model on z/OS with the outside world.

In this chapter we have seen that security administration can be done on z/OS, and distributed application components can reuse this information on distributed servers. We have also seen that security credentials can be propagated from distributed components onto the z/OS environment, and further auditing and authorization checking can be performed based on those credentials.



## End-to-end security scenarios

This chapter outlines several scenarios that can be evaluated for WebSphere on z/OS implementations using different kinds of security technologies. We focus on whether authentication and authorization is external or internal to z/OS, or a combination of the two. We give a high-level overview of the following scenarios:

- ▶ An HTTP server and WebSphere Application Server on z/OS in which authentication is done by the HTTP server.
- ▶ An external J2EE application authenticates through an SDBM LDAP interface on z/OS.
- ▶ An external authentication server authenticates natively to RACF on z/OS through an LDAP interface, then with an external authorization server.
- ▶ Expanding the security model further with back-end access.
- ▶ An authenticated user ID and credentials are sent from a distributed WebSphere Application Server to a WebSphere Application Server on z/OS.
- ▶ Use of a J2EE client application accessing a J2EE component directly on z/OS.
- ▶ Use of an incoming JMS message.

## 2.1 Simple HTTP with application server

The scenario illustrated in Figure 2-1 assumes that the user interface is inside a Web browser and that all transactions start with sending an HTTP request from the browser to an HTTP server. The HTTP server redirects all requests to WebSphere Application Server to access applications and data. The HTTP server and WebSphere running in a base configuration reside on the same z/OS system. The HTTP server takes care of authentication through RACF. Authentication and authorization to the application server are also done via RACF. The user's security credentials are propagated from the HTTP server to the application server. We assume that the user ID with which a user authenticates to the HTTP server is the same user ID that will be used to access the application server. The HTTP server sends the user's credentials (certificate, Basic Authentication headers) to the application server when making requests. In this case, the HTTP server only operates as a proxy server. It forwards credentials at the HTTP protocol layer level. There is no propagation of a user's identity but only propagation of the user's security credentials.

Using *basic authentication* (BA), the user ID and password in the HTTP header will be authenticated twice (at the HTTP server level and at the application server level). Using client certificate authentication, the client certificate will be forwarded by the HTTP server in an HTTP Private Header to the application server. Privacy is taken care of via SSL/TLS. Server certificates can be stored in either the file system or in RACF.

In this scenario, the use of the SAF EJBROLE and GEJBROLE classes would typically apply for J2EE role-based security. The User Registry setting in WebSphere would be Local OS.

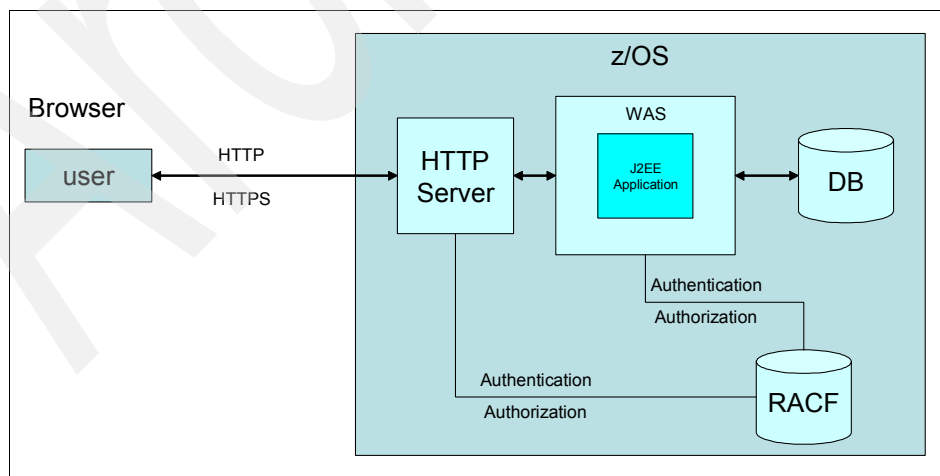


Figure 2-1 Simple HTTP with application server

## 2.2 Extended HTTP with application server and LDAP SDBM backend

This scenario (Figure 2-2) is similar to the previous scenario. However, the role of the HTTP server is different: it does not enforce authentication. This task is taken over by the application server. J2EE security forces the user to log on. The main difference between this scenario and the previous one is the use of LDAP with an SDBM backend. The main advantage of using the SDBM backend, which is tightly connected to RACF, is the possibility to connect distributed WebSphere Application Servers to the same registry as used by the application server on z/OS. This simplifies the environment by not having to synchronize registries. It also enables sites to use distributed application servers (on AIX, for example) as testing environments or front-end application servers connecting to the z/OS backends without having to set up different authorization paths or registry synchronization to secure access to back-end Enterprise Information Systems (EIS). SDBM provides a useful alternative to the local OS registry by expanding the usability range and is simple to set up.

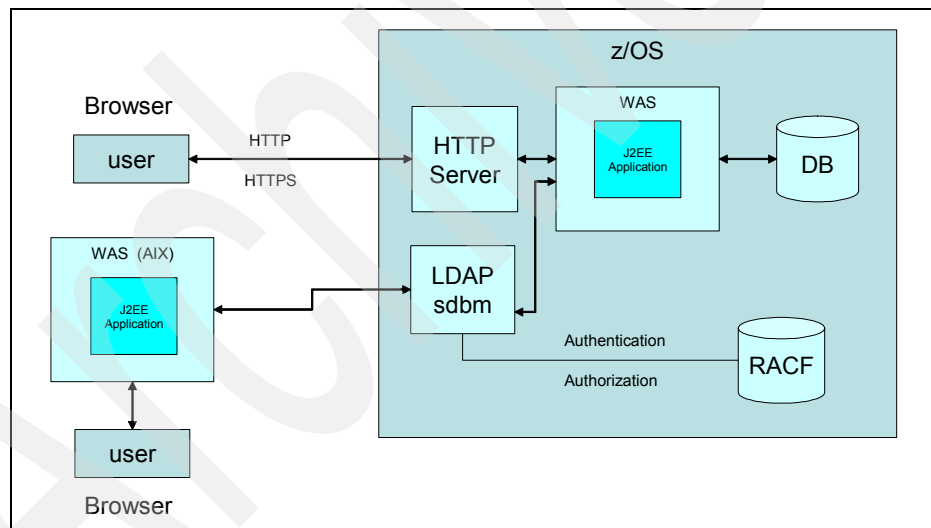


Figure 2-2 Extended HTTP with application server and SDBM backend

LTPA is the authentication mechanism to forward the user ID. The User registry setting in WebSphere would be LDAP. Encryption of transport between the browser and the HTTP server should be considered to guarantee privacy of data.

J2EE roles are mapped to SAF users or SAF groups in the applications' deployment descriptor (or at deployment time when using the administrative console of WebSphere Application Server to deploy applications). SAF users or

groups are validated at deployment time by the LDAP SDBM backend within RACF. Further access to back-end systems is handled by traditional SAF calls.

**Note:** LDAP SDBM support with WebSphere Application Server 6.0 has been made available as of level 6.0.2.5. For WebSphere Application Server 5.1, service level 5.1.222 is required.

## 2.3 External authentication and LDAP native authentication

The scenario illustrated in Figure 2-3 also assumes (as in the two previous scenarios) that the user interface is inside a Web browser and that all transactions start with sending an HTTP request from the browser to an HTTP server on z/OS. This scenario includes a separate authentication server running outside the z/OS platform, and a single HTTP server and application server running on z/OS.

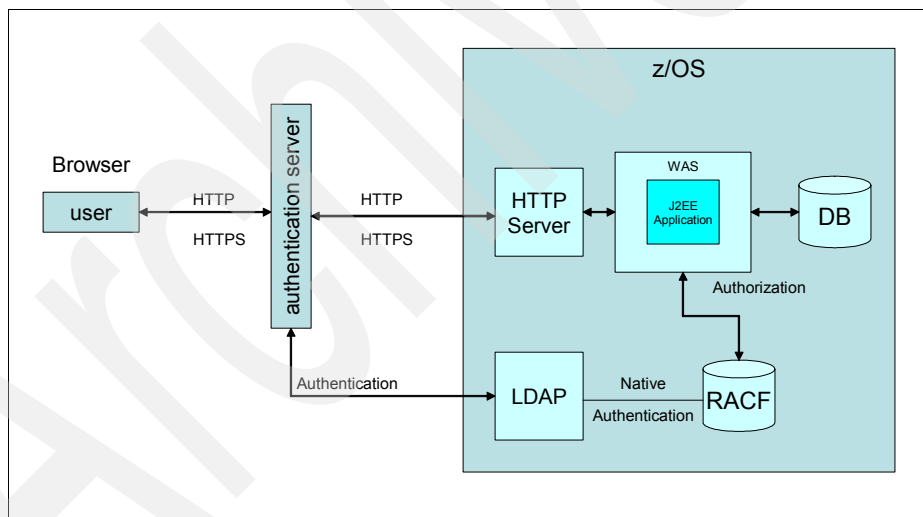


Figure 2-3 External authentication and LDAP native authentication

Authentication is initially handled by the authentication server. Each user request has to go through this authentication server first before going anywhere else. The authentication server's user registry is an LDAP server on z/OS. The LDAP server on its turn is loosely tied to RACF via the *native authentication feature* of LDAP on z/OS. This configuration makes it possible to authenticate outside z/OS using an RACF user ID and password. The HTTP server and application server

always receive an authenticated user ID from the authentication server in a format that they can understand. This user ID is used for authorization further on in the application server and is checked against RACF. There are actually two available mechanisms to forward the user ID:

- ▶ LTPA tokens
- ▶ HTTP headers and the *trust association interceptor (TAI)*

Privacy is implemented by using SSL/TLS between the browser and HTTP server. Between the HTTP server and the application server SSL could also be enabled.

## 2.4 External authentication, external authorization, and LDAP native authentication

This scenario, as shown in Figure 2-5 on page 18, has an external authentication and authorization server that both access LDAP on z/OS. LDAP is set up to use native authentication, which assumes that the user has a RACF user ID. In this case, the propagation of the user ID is done from the authentication/authorization server to the HTTP server running on z/OS and further on to the application server, also running on z/OS.

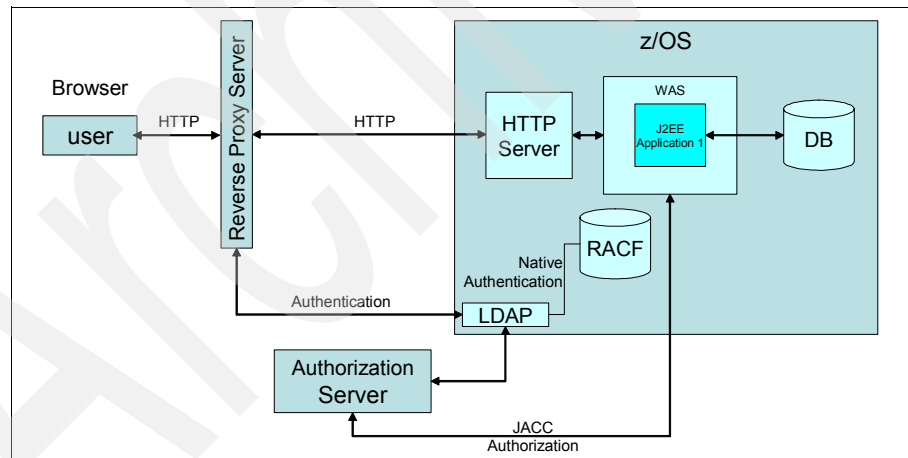


Figure 2-4 External authentication and authorization, and LDAP native authentication

Here again, the propagation of the user ID is done using two possible authentication mechanisms: LTPA tokens or the trust association interceptor (TAI). The model relies on the application server using the *Java Authorization Contract for Containers (JACC)* authorization model for authorization. Privacy is guaranteed similarly as discussed in the previous scenarios using SSL/TLS for the communication between the browser and the HTTP server.

## 2.5 End-to-end, authentication to authorization chain

This scenario, shown in Figure 2-5, works in much the same way as the model explained in 2.4, “External authentication, external authorization, and LDAP native authentication” on page 17. Here, however, we take into consideration how we authorize users to resources in the back-end system. In this scenario, we must propagate a single RACF user ID from the *reverse proxy server* to the HTTP server, to the Application server, and finally to the back-end system.

The back-end system requires an RACF user ID for authorization purposes. The front-end reverse proxy server requires an LDAP user registry. LDAP with native authentication and RACF is the perfect pair for authentication and authorization tasks throughout the entire flow. This couple makes it possible to use one single user ID from the authentication step at the reverse proxy server level, through the J2EE application server to the back-end system. Identity propagation between the reverse proxy server and the application server can be done using LTPA tokens or the trust association interceptor (TAI). Identity propagation between the application server and the back-end system is done using the local J2CA connectors and z/OS thread-level security (ACEE control block). Thread-level security is explained further in “z/OS environment: the big picture” on page 62.

**Attention:** The Authorization server is placed outside the z/OS platform in Figure 2-5, but it could also be placed on z/OS.

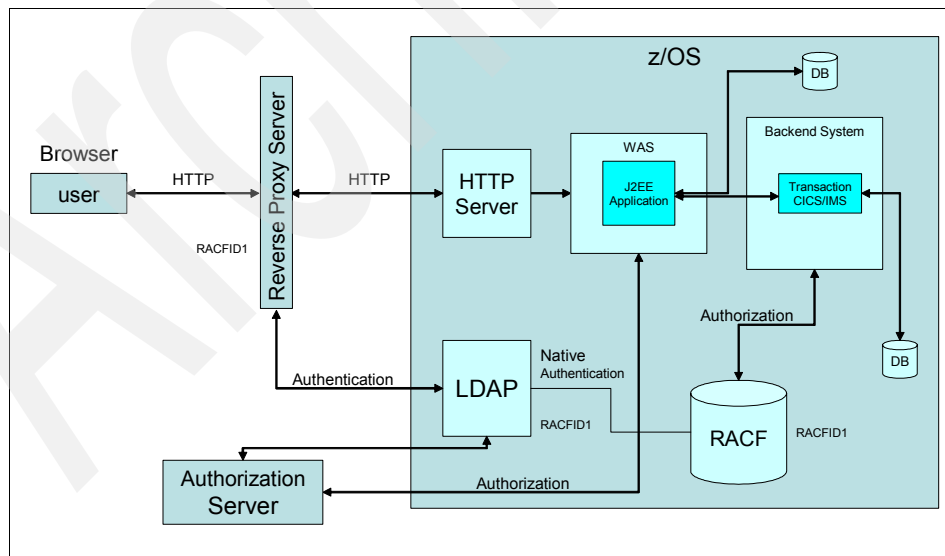


Figure 2-5 End-to-end authentication and authorization chain



## 2.6 J2EE client

This model, as shown in Figure 2-6, differs from the previous ones in that we use a J2EE client instead of a standard Web browser. The J2EE client communicates directly with the application server on z/OS using J2EE RMI-IIOP (Remote Method Invocation over Internet Inter-Orb Protocol). There is no HTTP server involved; therefore not all security functionality in the HTTP server is available. It becomes the client application's responsibility now to request authentication and will therefore directly communicate with the LDAP server on z/OS. The authorization part is done in the application server on z/OS.

There are two available protocols for communicating securely between a J2EE client and WebSphere Application Server for z/OS: *Security Attribute Service (zSAS)* and *Common Secure Interoperability (CSIv2)*. zSAS is an IBM protocol that is still available for backward-compatibility reasons. CSIv2 is a standard protocol that should be used for any new implementations. These protocols enable user ID propagation between J2EE environments. Using CSIv2, this user ID can be an LDAP distinguished name, a client certificate, a basic user ID, and so on. The user ID that is retrieved from the CSIv2 protocol can either be an RACF ID or be mapped to one. The transport layer can be secured using SSL/TLS.

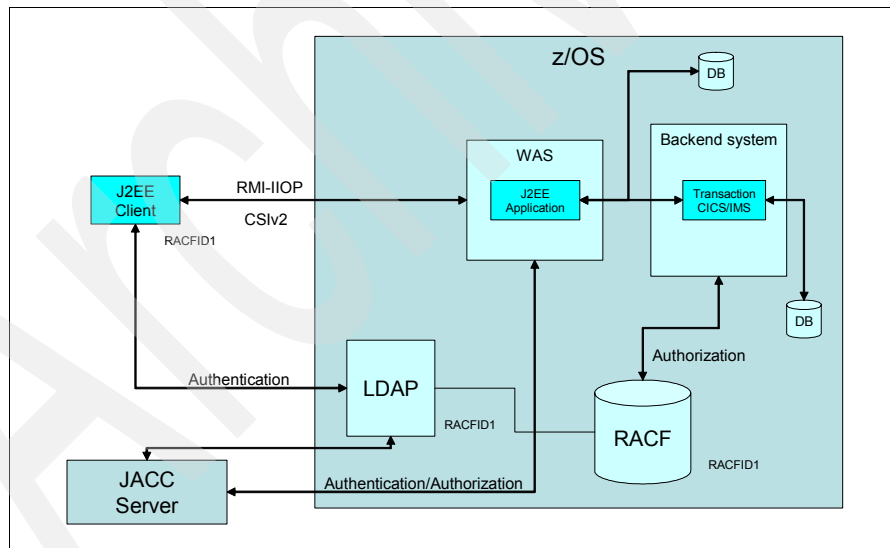


Figure 2-6 J2EE client

## 2.7 Bridged security between z/OS and distributed

At this point we describe a scenario that combines existing security infrastructures. The challenge for a large enterprise is to leverage their existing security mechanisms on all of their platforms and to create an end-to-end security context flow. For example, if WebSphere Application Server is used as a layer to give a large number of users access to data kept in DB2, important questions arise, such as: do we need to define all those potential users in RACF? If not, how do we secure access to DB2 or other back-end systems in a sufficient manner? Will we need a mechanism to synchronize all of our existing user registries?

The suggested scenario illustrated in Figure 2-7 describes a combination of distributed security making use of Tivoli Access Manager's authentication and authorization facilities and back-end accessibility checks in RACF on z/OS.

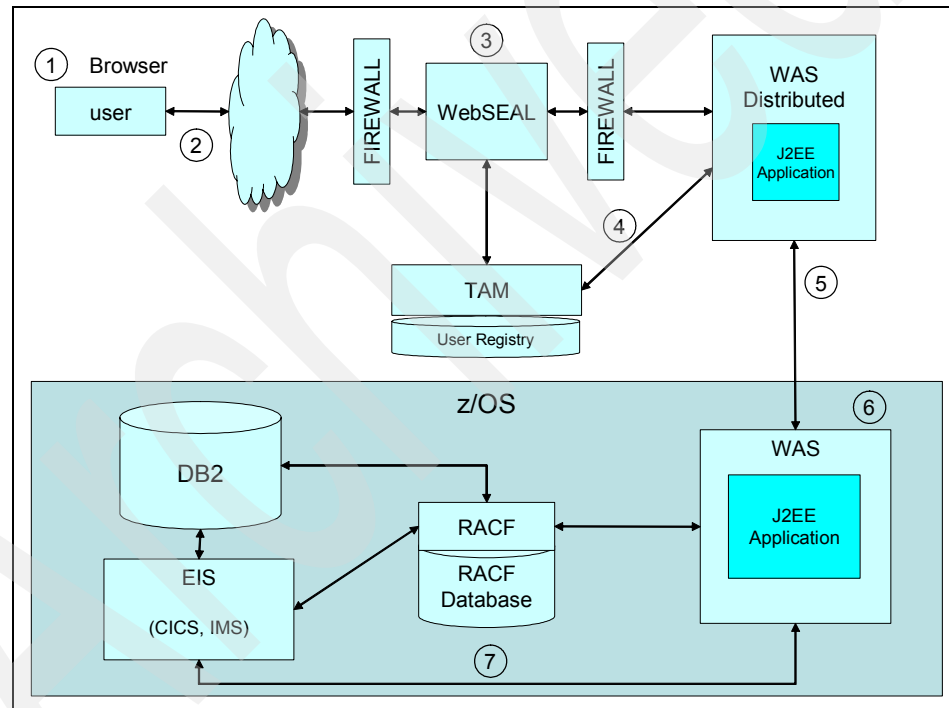


Figure 2-7 Bridged security

1. The user ID must be verified, based on user ID and password.
2. The message has to be encrypted to guarantee transport security, using SSL/TLS.
3. WebSEAL, which is a component of Tivoli Access Manager (TAM in Figure 2-7), authenticates the user and takes coarse-grained (URI) access

decisions. A Tivoli Access Manager - WebSEAL plug-in modifies the available userdatafield in the LTPA or TAI header and adds an entry (tag) depending on various things (net location of the browser request, for example) before encrypting them as an LTPA token. WebSEAL is positioned inside the DMZ. The tag will be mapped to, or is already, an RACF identity with sufficient authority to access z/OS back-end systems, thus providing a one-to-one or many-to-one mapping for back-end access. User registry is external to z/OS.

#### 4. WebSphere on distributed

All Web-related parts of the enterprise application are deployed here. Tivoli Access Manager is used to make J2EE role-based authorization decisions. Depending on Tivoli Access Manager's decision, the user will be allowed to access the application part (servlet, Enterprise JavaBeans or EJB™) that calls an EJB in WebSphere on z/OS (WebSphere Application Server on z/OS being the J2EE server and WebSphere Application Server on distributed acting as a client for J2EE).

#### 5. The user's credentials are sent over RMI-IIOP using the CSIV2 security protocol to WebSphere Application Server on z/OS.

#### 6. WebSphere Application Server on z/OS

A custom-built JAAS login module is configured to retrieve the tag from the credentials, map it to a system user, and set it as principal for downstream authorizations. WebSphere Application Server on z/OS is configured to use localOS as active user registry: role-based security constraint decisions are handled by the security server (RACF) on z/OS. EJBROLE/GEJBROLE profiles are checked for that purpose.

#### 7. Accessibility to z/OS backends is handled by RACF in the usual manner.

In this scenario there is no need to synchronize user registries: the addition of a tag to the user's credentials and the mapping of that tag to a z/OS system user is the key for back-end access on z/OS. The amount of mapped RACF IDs in this way can be relatively limited, depending on the different types of access you wish to give to the Internet user. For example, an unknown user coming in through the Internet could be given limited access to DB2 or already blocked at WebSEAL depending on enterprise policies, by assigning a default user ID and a corresponding limited-access tag. The next chapter elaborates on this scenario.

## 2.8 MQ client

Our main focus in this model, as shown in Figure 2-8 on page 22, will be on security-related issues regarding messages that trigger a Message Driven Bean (MDB). When a listener port passes on a message with a JMS or MQ format header to the MDB coupled to that listener port, WebSphere will not propagate

the user ID contained in the header. This of course seems to be valid, because no password is provided in the header and no full trust relationship exists.

Messages received asynchronously by the MDB are, in many cases, automated return messages from some kind of process, such as order received, handled, shipped, or delivered. In those cases it is unlikely that the message has been originated by a user application, or a user, but by a system or process. By default, the MDB analyzes the message and calls an EJB (stateless session bean), which will run under a system user. When the EJB is called, it might be desirable to run it under a user ID different from this system user ID, to be able to use standard J2EE role-based security at the user level and eventually propagate a user ID downstream that makes sense.

If you have this requirement, you can have the EJB run under a meaningful user ID by performing a JAAS login, with the JMSX user ID. This way the EJB can be scheduled with this user ID, and the user information from the MQ message can be propagated further.

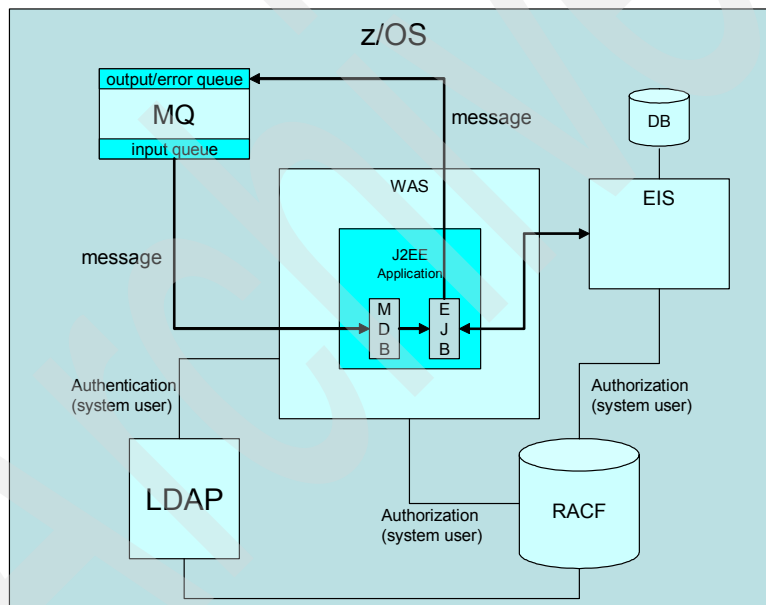


Figure 2-8 MQ MDB flow

The flow of this solution is as follows:

1. The MDB is triggered by the listener port in WebSphere.
2. Extract the (RACF) user ID from the message.
3. Generate a passticket for the RACF user ID, using special code that can generate the passticket.<sup>1</sup>

4. Call a JAAS login module with the (RACF) user ID sent in the message and the password and the passticket generated in the previous step.
5. Call the (stateless) session EJB with RUNAS set to the (RACF) user ID used in the previous step.

Authorization on the provider side, MQ in this case, should be thoroughly planned. The listener port that triggers the MDB has a one-on-one relationship with a queue as defined in the queue destination. MQQUEUE SAF class profile permits should reflect the identities related to the work being done after a message has been processed by an MDB.

---

<sup>1</sup> This code is available only as of z/OS Version 1 Release 7.



## z/OS and WebSphere security technology overview

This chapter describes a “broad brush” view of security concepts, standards, protocols, functions, and other artifacts that can be used in assembling an end-to-end integrated security solution when using WebSphere Application Server on z/OS. This chapter includes explanations about:

- ▶ General terms that we use throughout this chapter and this paper
- ▶ The differences between the “old world” and the “new world” and where the focus of security technology is now
- ▶ What areas should be focused on in an integrated solution
- ▶ Functions that are available in WebSphere Application Server for z/OS Version 6 in addition to the standard J2EE functions
- ▶ LDAP on z/OS

For more background information about J2EE security and RACF, refer to Appendix A, “J2EE security” on page 71 and Appendix B, “z/OS Security Server (RACF)” on page 87.

## 3.1 General concepts

In this section we explain a few general concepts and terms that we use throughout this chapter.

### 3.1.1 Single sign-on

Single sign-on (SSO) is a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where they have access permission, without the need to re-authenticate within a certain time period. Single sign-on reduces human errors, a major component of systems failure, and is therefore highly desirable but difficult to implement. More information about SSO can be found at:

<http://www.opengroup.org/security/sso/>

#### **When to look into this?**

When you are designing a solution in which a user wishes to access different applications and systems using a single authentication action.

### 3.1.2 Credential

A credential is reference information, such as a password or certificate, that can be used to authenticate a principal. The challenge is to have the credentials available at the place and moment when you actually need to perform the authentication. In the majority of cases, we imagine this as a user typing in an ID and password. However, there are more difficult cases in which the authentication or reauthentication must be performed by the system, such as when entering CICS. In that case, credentials must be available in the request.

#### **When to look into this?**

When you have a need to perform authentication or reauthentication at any point in the end-to-end solution.

### 3.1.3 Principal

A principal is an entity that can be identified and authenticated (for example, the initiator of a request, such as a user or requestor).



### 3.1.4 Security domain

A security domain is a scope that defines where a set of security policies are maintained and enforced (also known as a “security policy domain” or “realm”).

### 3.1.5 Subject

A subject is a set of principals and their credentials that are associated with a thread of execution. Principals and their credentials can be retrieved from the subject for various security functions such as authentication, reauthentication, authorization, and auditing.

## 3.2 The evolution of security integration

The emergence of Web-based applications has created new challenges for the protection of mainframe assets. Security was a key design point for the mainframe from the beginning, and over time it has evolved to become an extremely secure environment. Today, it acts as the central repository for sensitive corporate data within most large enterprises. Mainframes are both secure and persistent. They enable data to be hosted as long as it is needed and have well-defined, robust methods for access control.

New Internet applications require additional considerations:

- ▶ They open public access to sensitive mainframe resources.
- ▶ They often bring System z and non-System z models together in a collaborative way that leads to distinct security requirements.
- ▶ A multi-layer security implementation with the concept of trusted and untrusted zones (or DMZ) is often desired.

Given these concerns, many enterprises opted to form new organizations along with new I/T infrastructure to support their entrance into the world of the Web. As a result, many businesses selected distributed platforms as the foundation for their new Web infrastructure. In most cases, this approach was more complex and less secure. Typically the data required by a Web application resides on the mainframe where the infrastructure needed to protect the information already exists.

The challenge is to integrate distributed security with mainframe security to provide a unified protection scheme. Separating the application from the data

and spreading parts of the application across many distributed systems creates a complicated solution with direct impact on:

- Cost

Moving the application farther from the data affects the cost. Access to data is slower, additional hardware is required to handle marshalling and de-marshalling data exchanged between systems, different software licenses are needed, and administration costs increase commensurate with the increased complexity of the environment.

- Availability

The mainframe offers a “world class” high-availability environment. The design point of System z is “zero downtime” so there are inherent qualities in the core hardware and software to support availability. Moving access to mainframe data to the distributed platform can adversely affect availability because overall application availability is determined by the sum of the parts and, often, is only as good as the weakest element. Most mainframe users have invested in the hardware, software, and procedures that are required to support extreme availability. By placing new applications on the mainframe, this investment may be leveraged.

- Security

Besides the obvious concerns with operating system and application security on distributed systems, there is a more subtle effect. Most mainframe shops have created a security infrastructure that has been hardened and abides by a strict policy. Distributed application access to zSeries® resources mandates policy changes and additional security infrastructure to achieve the same level of confidence.

- Skills

A variety of new skills will be needed that can be difficult to acquire and maintain. Most often, the data will remain on z/OS, so skills are required in both distributed and mainframe security technology. In addition, skills specific to cross-platform security considerations are required.

The challenges outlined above suggest that a different approach is desirable. Most of impacts can be eliminated or reduced by implementing the new Web applications within the existing mainframe infrastructure.

### 3.2.1 Classic view

Figure 3-1 on page 29 shows a high-level common infrastructure view of the mainframe environment, the traditional approach to interaction with external users focused primarily on members of the enterprise. Secure access to mainframe resources via traditional interfaces is well understood and effective.

Our challenge is to leverage this robust infrastructure to support Web interfaces and users that are not necessarily members of the enterprise.

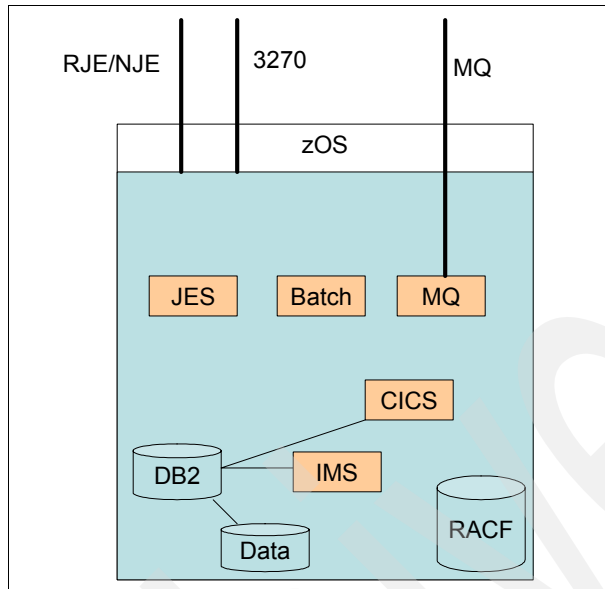


Figure 3-1 Common environment

Adding to this infrastructure has several advantages:

- ▶ There is no need to build a new infrastructure that will require new skills.
- ▶ The user directory can be reused to allow existing users to have access to new services automatically. RACF user IDs and passwords or associated digital certificates may be used for Web access. When logging in to an application, the user does not have to be worried about logging into a back-end system (such as CICS or IMS). A single security manager controls both the Web application and the associated resource managers, so secure identity propagation can be achieved very efficiently.
- ▶ The security infrastructure is maintained. RACF provides for separation of security concerns from application concerns. Identity management, audit policy, and trust associations may be controlled by security personnel independent of developers or Web infrastructure administrators.
- ▶ Access to data is most efficient because the data is often already on the mainframe. With Web applications and associated resource managers such as CICS, IMS, and DB2 operating in a shared memory environment, cross-memory interaction provides a secure, high-performance pipe between resources, avoiding the latency and additional security exposures of network connections.

- ▶ Data integrity is improved by leveraging a high-performance two-phase commit coordinator built into the operating system. Cross-memory program calls to resource managers avoid the complication of waiting for time-out thresholds to be detected in order to identify and begin recovery of a failure situation.
- ▶ Management is simplified because the same staff can handle the new service and all tools can be reused. In the mainframe, auditing and administration are usually highly developed. Keeping the business functions on the mainframe maintains the processes that are already in place.
- ▶ New people and infrastructure are not required. Combined with reduced complexity, this saves considerable cost. Existing hardware can usually be utilized in the beginning without additional investment. As the Java workload grows, the zSeries Application Assist Processor (zAAP) can be employed to dramatically lower the cost of new workloads. Additionally, software cost (OS, middleware, and so on) for the distributed system is avoided. Some WebSphere software costs can also be reduced because multiple instances of WebSphere supporting different applications or different roles, such as development test, quality assurance, and training, can run on a single LPAR and share physical resources. Virtualizing server instances while maintaining the necessary isolation provided by the z/OS address space structure eliminates the need to purchase and maintain redundant infrastructure.

### **Current security in this environment**

RACF is securing subsystems and applications running in this environment. Users are granted specific authority to the resources that they need. This is frequently done by associating users with functional groups. To leverage this security system, the system architect should look for ways to enable users to use these resources externally while logged in with their RACF identity. The remainder of this chapter mentions methods to enable this functionality.

## **3.2.2 The introduction of WebSphere Application Server**

In the simplest case, the first step is usually to create an application that can “front-end” the enterprise data and open up access to Internet and intranet users. The goal is often just to simplify use of mainframe applications by allowing access with more comfortable ways (PC clients or Web browsers). Figure 3-2 on page 31 demonstrates, at a high level, what the base infrastructure picture looks like when WebSphere Application Server has been added. This picture also includes an HTTP Server, CICS Transaction Gateway (CTG), and IMS Connect. They were included for completeness because they are a common additions when introducing an application server into this environment. CTG and IMS Connect deliver the necessary connectivity between the Java Web application and CICS or IMS transactions. The connection is transactional, secure, and

optimized by support for cross-memory calls. RACF controlled access to CICS and IMS allows for secure propagation of user identity extending authorization control to the back-end system.

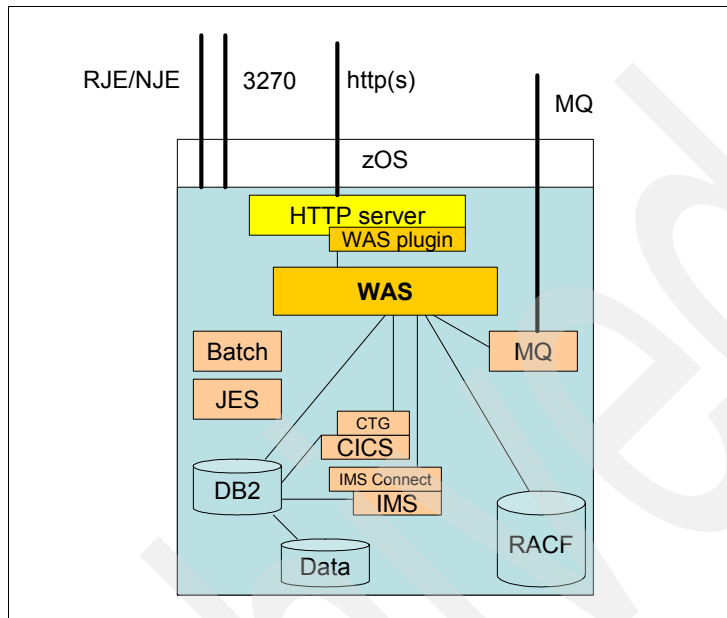


Figure 3-2 WebSphere Application Server is introduced

Before the introduction of WebSphere Application Server, security was probably a comfort zone in the enterprise. Policies exist and protection is understood. Now, there are new channels that have to be secured. Also, some decisions must be made. Can any Internet user access the Web application? When authenticated, does the Web user's identity have to be used when accessing the existing data or transactions? Is there any information that will require encryption?

**Important:** This book assumes that RACF (z/OS Security Server) is the security product of choice in the environment. Any SAF-compliant security software should function, but examples will have to be edited to apply to the specifics of that software.

A few choices have been made in Figure 3-2 that might not apply to every environment. Also, some choices are needed that might not be demonstrated by the high-level view provided by this figure. Table 3-1 on page 32 lists these choices with a description of the effects of specific decisions. The list is not completely inclusive; it exists to provide high-level guidance. See Chapter 2,

“End-to-end security scenarios” on page 13 for further explanation of how to make the proper choices.

*Table 3-1 Choices when introducing WebSphere Application Server*

Choice	Effect
Should Web content be served initially from an HTTP server (pictures) or a WebSphere HTTP listener?	An HTTP server is often placed in front of the application server for two primary purposes. First, when dealing with the Internet, the browser level employed by the user might not be known. The HTTP server has learned to deal with the nuances of different browser levels over time. The transport listener, on the other hand, is specifically designed for standards compliance. Second, the HTTP server supports the Fast Response Cache Accelerator, which optimizes delivery of static content by allowing it to be delivered directly from the TCP/IP stack. An additional benefit of placing the HTTP server on z/OS is the ability to point the HTTP server at the static content directory created by WebSphere during application deployment. Hence, there is no need to manually populate a file directory, which eliminates a deployment step.
Does the user ID have to be used to access the backend?	Often, in a distributed application server environment, a single trusted user identity is used to access mainframe resources. When the actual user is authenticated, an alias identity (frequently the server identity) is used for access to mainframe resources. This means that the mainframe resources trust the application server, and specific user access control decisions are delegated to the application server. In some highly secure environments, it is important to employ security constraints in the back-end resources. This means that the actual client credential has to propagate to the back-end resource, which results in performance and administrative considerations when the application server is not deployed on z/OS.
Which back-end systems should WebSphere use?	Figure 3-2 on page 31 shows WebSphere Application Server connected to every back-end system in the pre-WebSphere Application Server environment. CICS Transaction Gateway and IMS Connect have been added to the view of the installation because they provide the implementation of the J2EE Connection Architecture used by Web applications to call Enterprise Information Systems as defined by the Java Enterprise Edition specification. See the <i>WebSphere for z/OS V6 Connectivity Handbook</i> , SG24-7064 for more about setting up and architecting these connections.

From a security standpoint, we need to decide some things when setting up this environment:

1. What user registry will be used? Will an SAF user ID be required and checked by setting up a “local OS” user registry, will we use LDAP, or will some sort of custom user registry be set up? See 3.4.3, “User registry” on page 42 for more details.
2. Once authenticated, will J2EE security be used and to what extent? See 3.4, “Related security technology” on page 41 for a review of this security.
3. How will we access our data and legacy applications? Do we need each user defined? How does this affect our infrastructure (SAF identities for each user)?

At this point, our goal is simply to identify these questions. After reading the infrastructure examples, the effect of these choices will be better understood. An example includes the user registry. An LDAP user registry might become a desirable choice because it can enable a single sign-on solution in environments with a heterogeneous collection of servers. It can also force some actions on the backend because we might need to turn the LDAP credential into an SAF credential that is understood by the back-end resources. We will explore various options for identity mapping in a later section.

### **3.2.3 Further hardening through a reverse proxy**

At this point, the environment exists where Web users can use an application running on z/OS without knowing that they are accessing a mainframe. Data can be accessed that was present before this application entered the picture, and the workload does not have to leave the system. There are additional things to consider: For example, how can user access be limited to getting into WebSphere Application Server at a coarse-grained level? How can exposure of IP addresses, ports, and so on be avoided while still allowing access to the application? Figure 3-3 on page 34 introduces the concept of a reverse proxy and firewalls to the environment.

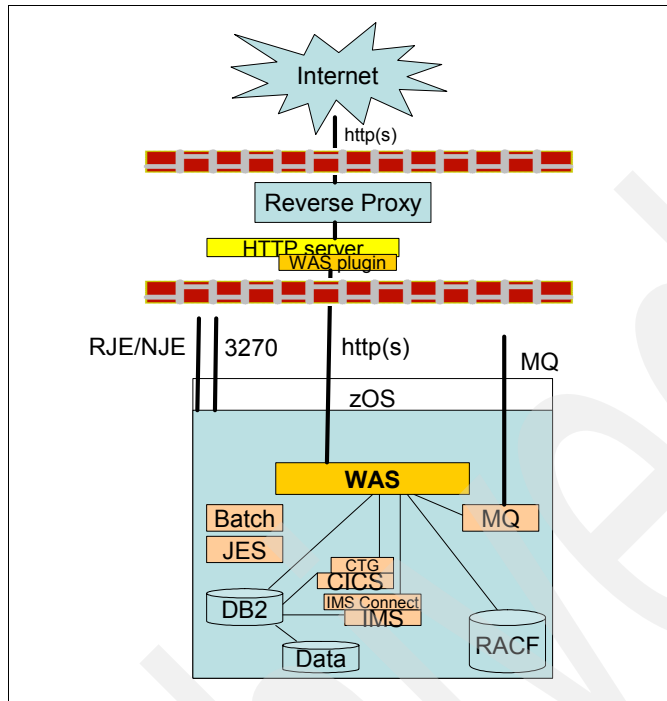


Figure 3-3 Infrastructure with proxy

One of the most important and common infrastructure choices introduced in this picture is the demilitarized zone (DMZ). Represented by the area between the firewalls, this zone isolates the Web servers from the application servers and establishes a boundary between the trusted and non-trusted networks. This internal network is isolated from the Internet by a firewall, which is responsible for protecting the Web server (or, more likely, Web servers) from attacks. This firewall protects the Web server from denial of service attacks as well as viruses and other threats bent on keeping people from accessing the Web server. This firewall also does Network Address Translation (NAT). NAT hides the addresses in the internal network from the addresses on the Internet. The Internet might see the Web server at address 202.123.64.22 while the DMZ network has the server at address 10.9.5.3. Because the Internet cannot access the server by its local address, the Web server can easily separate requests that come from the Internet from requests that come from the internal network.

The second firewall is a bit more sophisticated. It is responsible for protecting the internal secure network (intranet) from attempts to gain access to critical resources. This firewall only admits verified users with valid requests.



Another architecture choice that was made was the use of the reverse proxy. With this proxy the real application server can be hidden from the public. This is important and creates a more secure environment. More important, having a reverse proxy inside the DMZ enables us to move out some authentication and authorization before entering the trusted zone.

As if this writing, no proxy server is available that integrates directly with RACF, so to perform authentication and some authorization at the proxy, another user registry must be introduced. Figure 3-4 shows a link to LDAP for user information.

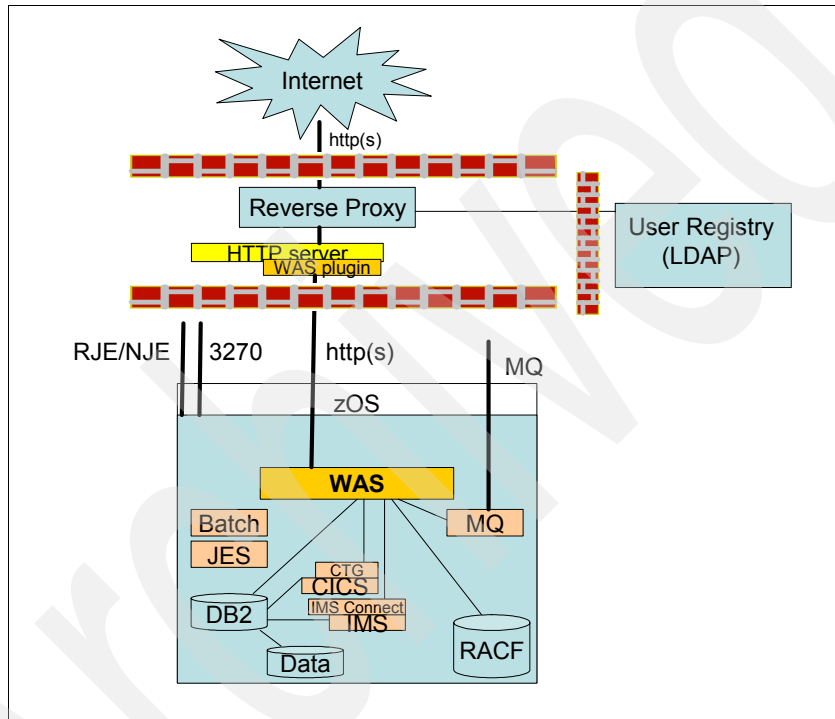


Figure 3-4 Infrastructure with LDAP

As with the other infrastructure changes, this raises an additional set of questions. LDAP will store the user information, so how does an RACF ID become the credential used for accessing the legacy systems? Will RACF need to somehow synchronize with LDAP? Do we just want to use LDAP on z/OS as a front-end to RACF (SDBM)? In Chapter 4, “A sample solution” on page 53, we describe one of the possible scenarios in considerable detail.

### 3.2.4 Beyond HTTP and HTTPS

For simplicity, the discussion up to this point has centered on HTTP access to applications. Security must also be concerned with any other channel that could be used. Figure 3-5 has added JMS and RMI.

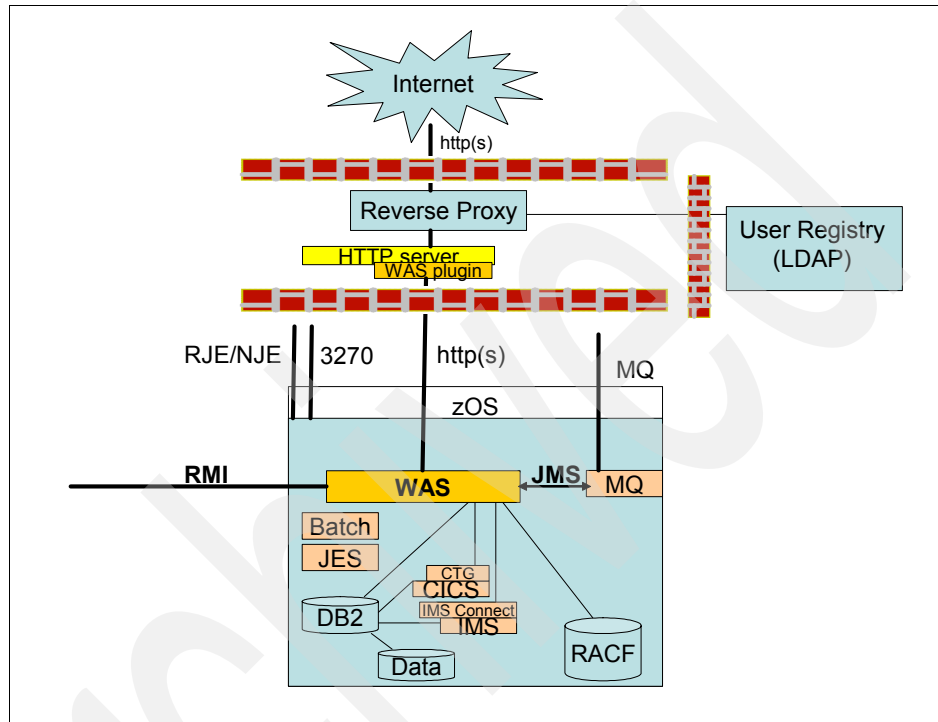


Figure 3-5 RMI and JMS channels

#### RMI

The EJB container performs authentication when it receives a request for an EJB method by way of RMI-IIOP. You can configure the authentication mechanisms that EJB container supports. This is specified as part of the server definition. Of course, the capabilities of the client sending the RMI-IIOP request and the location of the client with respect to the J2EE server control which of the specified methods is used.

The EJB container supports the following means of authentication:

- ▶ Basic authentication supporting user ID and password
- ▶ Certificate-based authentication supporting X.509 certificates within the context of a public key infrastructure

- ▶ Kerberos authentication
- ▶ Asserted identity
- ▶ Unauthenticated

## JMS

Messaging security operates as part of WebSphere Application Server global security and is enabled only when global security is enabled. If so, user IDs requesting connections to the JMS provider are authenticated, and can then be used by the JMS provider to control access to its resources such as queues. The user ID that will be used for authentication can be provided by the application or the container, and depends on a combination of settings. When the authentication fails, the connection request is rejected.

Beyond simply deciding whether the container or application will handle the authentication, a variety of other decisions are needed at this point. Table 3-2 highlights the most important of these.

Table 3-2 JMS choices

Question	Options
Which messaging provider should be used?	<p>With WebSphere 6.02, there are several choices for providers:</p> <ul style="list-style-type: none"> <li>▶ Default Messaging, also referred to as WebSphere Platform Messaging or the Service Integration Bus</li> <li>▶ WebSphere MQ Messaging, including support for shared queues</li> <li>▶ Generic Messaging to support third-party message providers</li> </ul> <p>Each provides unique security concerns and choices. See <i>WebSphere for z/OS V6 Connectivity Handbook</i>, SG24-7064, for more information.</p>
What security setup is needed?	Depending on the provider choice, roles in WebSphere might have to be configured or SAF profiles could be used to grant some specific access.
How will message confidentiality be achieved if it is needed?	Messages can contain a large variety of information, so it is usually a good idea to prevent messages in transit from being opened or modified by any unauthorized means. <i>WebSphere for z/OS V6 Connectivity Handbook</i> , SG25-7064, can provide significantly more details.

### 3.2.5 Advanced security concerns

As the environment grows and the number of application servers increase, the requirement for one logon to all applications or services a person has access to starts to increase in importance. This is normally implemented with one logon to the proxy server. Trust relationships between the proxy and the application servers are established to avoid separate logon to each back-end server.

This type of functionality is found in access manager products such as IBM Tivoli Access Manager. The demand for single sign-on might cross enterprises. Then additional products will be required to synchronize user registries between platforms. Tivoli Federated Identity Manager (TFIM) can be used to implement single sign-on according to standards such as SAML and Liberty.

In the generic SSO case, the user identity must be available in a form that is accessible to all participants in the SSO environment. This is most commonly achieved through the industry-standard LDAP registry. For this reason, WebSphere SSO requires an LDAP registry. (Technically it also supports a Custom Registry but because the front-end HTTP server only supports an LDAP registry, LDAP is the desired choice. If the user identities were only available in RACF, you could leverage the LDAP interface into RACF for SSO.

Introducing an access manager into the infrastructure requires the users to be defined in the access manager's directory. In Tivoli Access Manager this is an LDAP server. The WebSeal component of Tivoli Access Manager can act as a reverse proxy, enabling us to maintain the architecture outlined in 3.2.3, "Further hardening through a reverse proxy" on page 33 while extending it to support single sign-on. If the users are defined only in RACF at this point, it would be necessary to either re-define to users in the access manager database or use the LDAP interface into RACF.

If the applications servers all reside on z/OS, WebSphere SSO can use RACF. WebSphere SSO requires that all servers exist in the same domain (for example, mycompany.com), that all servers share the same registry for authentication, and that cookies are enabled so the encrypted LTPA token might pass back and forth between the server and the client. For cases where multiple cells on different platforms want to participate in SSO, there is an additional requirement. The symmetric key used to generate the LTPA token must be available in the keyring of each server in each cluster. Absent a shared key, there would be now way for a server in cell 2 to validate the token sent from a server in cell 1. For homogenous z/OS environments, sharing key rings is straightforward. In the heterogeneous case, the propagation of keys to each environment must be performed.

Figure 3-6 on page 39 shows the infrastructure with the components required for single sign-on. Note that the picture is not much different from the picture we had when authentication was moved out to the proxy. We have added the Tivoli

Access Manager product to handle authentication and some coarse-grained authorization. The policy server is required for administering policy information. For example, it would be used when creating new users or protecting resources. Tivoli Access Manager uses LDAP as the user registry only. The access control database is a proprietary implementation, and because of this, the policy server is needed to manage it.

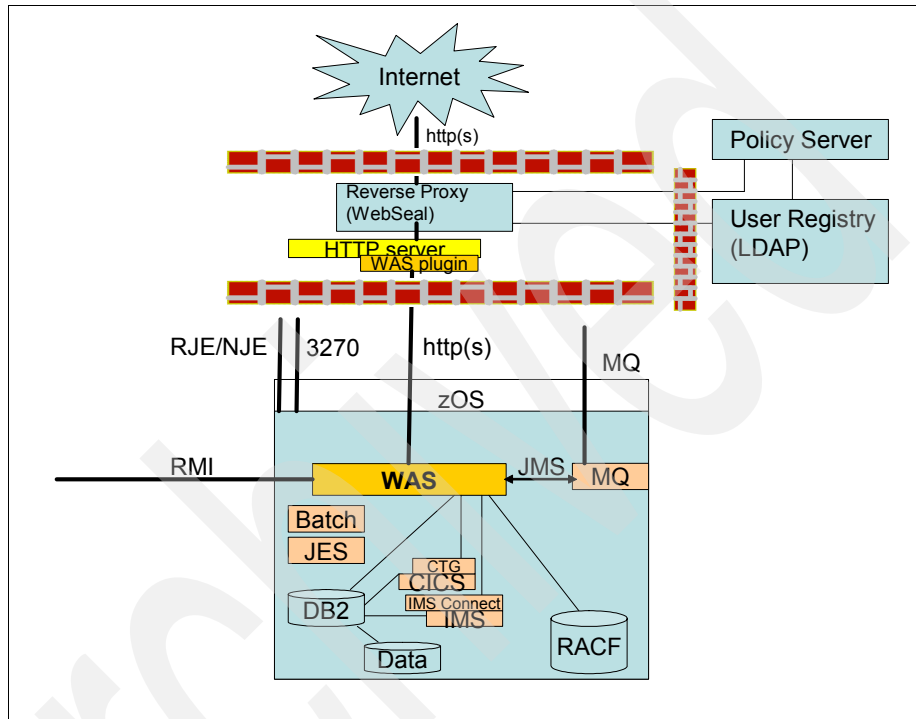


Figure 3-6 Infrastructure with SSO

More than ever before, planning access to the legacy applications and data becomes an important consideration. The ID from LDAP can be used with some setup assuming it is also in RACF. Keeping these registries in sync becomes a necessity if the desire is to have the end user's ID flow through to the final data access control.

If you need to keep the same users in both RACF authentication and the access manager, the best choice for storing access manager data is the LDAP server on z/OS. Figure 3-6 shows LDAP outside the scope of z/OS. This was done only for visualization purposes. Actually, running LDAP and RACF on the same LPAR makes it easier to establish robust solutions to synchronize data in the access manager database and RACF. In addition, LDAP on z/OS has a feature that makes it possible to use the same password for both the access manager and RACF.

### 3.3 What are the (new) security points of interest?

Figure 3-7 provides a high-level overview of WebSphere Application Server in a z/OS paradigm. WebSphere Application Server can exploit the security features of z/OS, as well as Java security features, HTTP-related security functions, and back-end system security functions.

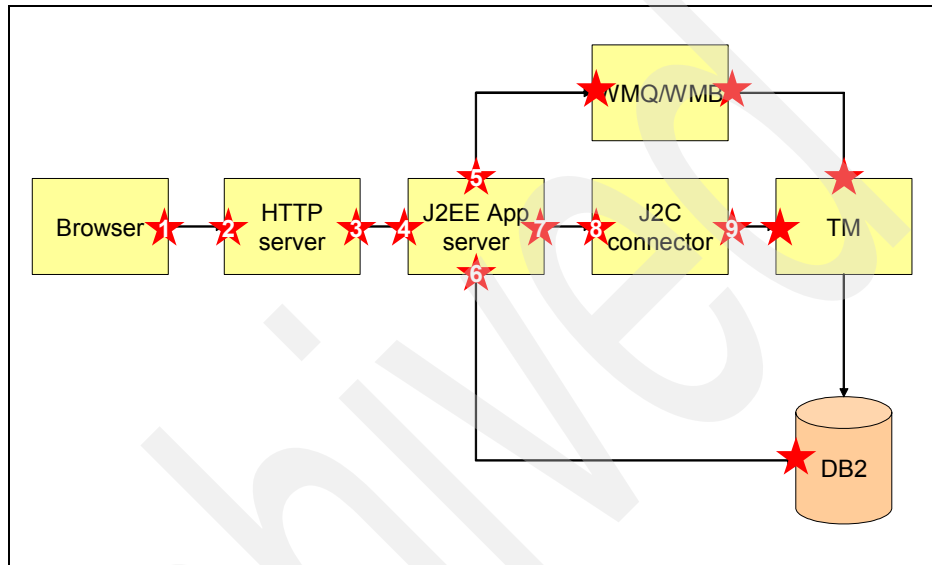


Figure 3-7 High-level view of WebSphere Application Server, RACF, and back-end systems

A security solution is only as strong as its weakest component, and the security design for a WebSphere installation and its applications should be end-to-end. The major area of interest, though, is where the request comes into z/OS, and in some cases directly into WebSphere Application Server. Of second interest is how WebSphere Application Server will be integrated with back-end systems from a security point of view. The J2EE security functions that are available in WebSphere Application Server can also be interesting, especially for authorization purposes.

Referring to the diagram in Figure 3-7, you might need to take the following items into account for the design of the solution:

1. When leaving the browser:
  - SSL for data privacy (encryption) and data integrity (message digest)
  - The use of certificates

- Transmission of user ID and password in the BA header of the HTTP request
- 2. When entering the HTTP server on z/OS:
  - SSL for data privacy (encryption) and data integrity (message digest)
  - Certificates
  - (Custom) user exits for authentication
  - Configuration settings to use LDAP or RACF
- 3. When leaving the HTTP server on z/OS:
  - Configuration settings for propagation of user ID and credentials
- 4. When entering WebSphere Application Server on z/OS:
  - J2EE settings for authorization and further propagation of identity and credentials
- 5. When going to a WebSphere MQ queue manager:
  - Security configuration settings in WebSphere for authentication and configuration of user ID and credentials
- 6. When going to DB2 on z/OS:
  - Security configuration settings in WebSphere for authentication and configuration of user ID and credentials
- 7. When going to a back-end system using a J2C connector on z/OS:
  - Security configuration settings in WebSphere for authentication and configuration of user ID and credentials
- 8. Connector-specific authentication settings
- 9. Connector-specific settings for propagation of user ID and credentials to back-end system

The following sections provide an overview of the security-related functions and technologies available in J2EE, WebSphere Application Server, LDAP, and z/OS.

## 3.4 Related security technology

This section provides an overview of additional WebSphere Application Server-specific standards, protocols, and functions that can play a role in any security design for WebSphere Application Server on z/OS and or its applications. You can use this as a check list for the architecture.

### 3.4.1 LTPA

Lightweight Third Party Authentication (LTPA) is intended for distributed multiple application server and machine environments, including z/OS and other platforms. LTPA generates a security token for authenticated users that can be used with multiple servers. Considerations for LTPA are:

- ▶ LTPA supports single sign-on (SSO).
- ▶ LTPA requires that the configured user registry be a centrally shared repository such as RACF or LDAP or that the involved registries are synchronized.
- ▶ LTPA requires a symmetric cryptographic key that is shared among all of the servers in a domain.

#### **When to look into this**

When you are designing a solution in which the first authentication is performed on a distributed server, before accessing WebSphere Application Server on z/OS.

### 3.4.2 REXX executables to add RACF definitions

The WebSphere Application Server BBOSBRK executable implements the required RACF definitions in order to run WebSphere Application Server for z/OS Version 6 on z/OS. The process will guide the user to create pertinent user IDs, groups, and resources. However, this is only one part of the entire security setup for WebSphere Application Server. WebSphere Application Server-specific entities (for example, J2EE roles) still might have to be defined in J2EE applications. Both components, internal security within WebSphere Application Server and external security using RACF, provide the complete security implementation for deployment in the z/OS environment.

### 3.4.3 User registry

User registry is a key component of applications. It contains information about users, identities, and groups for security domains. In WebSphere Application Server, a user registry authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization. There are several types of user registries, including:

- ▶ LDAP (Lightweight Directory Access Protocol) server
- ▶ Database-based user registry
- ▶ Operating system's user registry
- ▶ File-based user registry



The most common user registry today is the LDAP directory, or other user registries supporting LDAP to query users (for example: Domino® directory). User registries have different methods for organizing users in the registry. Generally, users and groups are mapped to some type of hierarchy in the user registry, possibly following the company's organizational structure.

### **LocalOS**

Using LocalOS as the active user registry for WebSphere Application Server on z/OS indicates that all authentication and authorization checks in WebSphere Application Server are done directly against RACF. The SAF EJBROLE and GEJBROLE classes are used to check access to the J2EE roles defined in the deployed Java Web applications.

**Note:** The name EJBROLE might be confusing. Besides EJBs it also applies to security constraints put on a servlet or JSP™ by means of a J2EE role.

### **LDAP**

The Lightweight Directory Access Protocol is a subset of the Directory Access Protocol and the X.500 OSI directory service. It is the directory service of choice for the Internet, providing a rich variety of features to support any kind of information or applications. It supports open standards and contains well-documented, well-known, and easy-to-use APIs.

### **Custom registry**

WebSphere Application Server allows for the special definition of a custom user directory. A custom user registry is a customer-implemented user registry that implements the UserRegistry Java interface as provided by WebSphere Application Server. A custom-implemented user registry can support virtually any type or form of an accounts repository from a relational database, flat file, and so on. The custom user registry provides considerable flexibility in adapting WebSphere Application Server security to various environments where some form of a user registry, other than Lightweight Directory Access Protocol (LDAP) or LocalOS, already exist in the operational environment.

## **3.4.4 Global security**

Global security is the “master switch” in the administrative console of WebSphere Application Server. When Global security is off, there is virtually no security inside the scope of WebSphere. If it is switched on, all settings related to security (such as the user registry choice, LTPA as authentication mechanism, and so forth) will come into effect.

## 3.5 LDAP on z/OS

In this section we present the most important definitions in an LDAP environment.

### 3.5.1 Distinguished names

*Distinguished names* are LDAP objects that identify a subject defined in the LDAP directory or user registry. A distinguished name is a unique name for an entry in the directory service.

### 3.5.2 Native authentication

LDAP *native authentication* provides for security checking against RACF user IDs and passwords. It requires a TDBM backend and DB2 tables. It is used when control is needed through a user registry or for implementation of single sign-on (SSO). It also may be used when RACF user IDs and passwords need to be reused using a LDAP interface. In addition, native authentication can be used for front-ending WebSphere Application Server on z/OS with RACF.

### 3.5.3 User registry

Lightweight Directory Access Protocol (LDAP) is a user registry in which authentication is performed using an LDAP binding. WebSphere Application Server security provides and supports implementation of most major LDAP directory servers, which can act as the repository for user and group information. These LDAP servers are called by the product processes (servers) for authenticating a user and other security-related tasks (for example, getting user or group information). This support is provided by using different user and group filters to obtain the user and group information. These filters have default values that you can modify to fit your needs. The custom LDAP feature enables you to use any other LDAP server (which is not in the product supported list of LDAP servers) for its user registry by using the appropriate filters.

To use LDAP as the user registry, you need to know a valid user name (ID), the user password, the server host and port, the base distinguished name (DN) and if necessary the bind DN and the bind password. You can choose any valid user in the registry that is searchable. In some LDAP servers, the administrative users are not searchable and cannot be used (for example, cn=root in SecureWay®). This user is referred to as WebSphere Application Server security server ID, server ID, or server user ID. Being a server ID means a user has special privileges when calling some protected internal methods. Normally, this ID and password are used to log in to the administrative console after security is turned on. Other users who are part of the administrative roles may be used to log in.

## 3.6 Administration

In this section we discuss some important information regarding administration.

### 3.6.1 Administrative console

The administrative console of WebSphere Application Server for z/OS Version 6 implements WebSphere Global Security and is controlled and maintained by the WebSphere administrator. This person may not be the installations' security administrator, because the WebSphere administrator is concerned primarily with internal WebSphere activity. However, the WebSphere administrator should work in tandem with the Security administrator in regard to defining, controlling, and authorizing users and groups to J2EE roles.

### 3.6.2 z/OS Security Server (RACF)

The z/OS Security Server (RACF) security administrator implements user profiles, dataset profiles, general resource profiles, and system settings. In addition, some installations might allow the security administrator to define audit settings normally reserved for system auditors.

RACF administrative authorities fall into multiple categories:

- ▶ Security administrators can control the entire environment (centralized) or a limited portion of the RACF structure (decentralized).
- ▶ Auditors can define audit levels for successful and unsuccessful accesses on user IDs, on resource profiles, or at the system level by class and level of access.
- ▶ Data or storage administrators can be given authority to access all data on the system and define profiles in specific RACF classes.

Users obtain permission and accountability in RACF by the assignment of a unique user ID. Each user ID is validated for authenticity when the user provides a valid password.

Access to resources can be to either z/OS datasets or other resources such as CICS transactions, programs, J2EE roles, or DB2 objects.

When you combine an application, such as z/OS HTTP Server, with middleware that supports a secure protocol, such as SSL, and the secure certificate management functions of RACF, you can implement a secure certificate

environment on z/OS. For implementation details about setting up z/OS HTTP Server and SSL, see the following references:

- ▶ *z/OS HTTP Server Planning, Installing, and Using*, SC34-4826
- ▶ *z/OS Cryptographic Service System Secure Sockets Layer Programming*, SC24-5901

### 3.6.3 LDAP

z/OS Security Server (RACF) provides definitions of users and groups, as well as access control for resources. RACF can act as a user repository for LDAP setup in Native Authentication or SDBM. You can set up either method to:

- ▶ Add new users and groups to RACF
- ▶ Modify RACF information for users and groups
- ▶ Retrieve RACF information for users and groups
- ▶ Delete users and groups from RACF

LDAP is used by Tivoli Access Manager as a user registry that can be synchronized with RACF via the IBM Directory Integrator (IDI) (if LDAP is not in native authentication or SDBM mode).

The SDBM database of the LDAP server implements portions of the ADDUSER, ADDGROUP, ALTUSER, ALTGROUP, DELUSER, DELGROUP, LISTUSER, LISTGRP, CONNECT, REMOVE, and SEARCH RACF commands. An individual user has the same authority through SDBM as with normal RACF commands. The SDBM database of the LDAP server makes use of the R\_Admin “run command” interface to accomplish its access to RACF data. As a result, this support is subject to the restrictions of the R\_Admin interface. One restriction in particular affects return of search results.

The SDBM database allows for directory authentication (or bind) using the RACF user ID and password. The RACF user ID must have an OMVS segment defined and an OMVS UID present. The RACF user and group information that make up an identity can be used to establish access control on other LDAP directory entities. This expands use of the RACF identity to the rest of the LDAP-managed namespace.

### 3.6.4 Naming conventions

A good practice for z/OS installations is implementing easy-to-understand and flexible naming conventions. User IDs, datasets, and general resources should have names that will be meaningful to application developers, management, and security personnel in order to enhance the z/OS security environment.

### 3.6.5 Practices

Audit-ready practices should be implemented as management guidance for implementing security. Accountability of human and non-human users of the system should be provided. Accessibility by users to resources should be available and documented. Regular audit reviews and settings should be implemented. Administrative control should be delegated and separated in order to maintain separation of responsibilities.

### 3.6.6 Procedures

Any user who can manipulate and control security settings should follow a well-defined set of documented procedures that detail the steps of accomplishing various job flows and tasks.

### 3.6.7 Architectural suggestions

Security should be incorporated into applications as early as possible during requirements definition. WebSphere users must be identified to specific RACF user IDs. J2EE roles should have meaningful names. RACF groups are defined for users with similar access requirements for access to J2EE applications. If “Local OS” is defined, RACF EJBROLE and GEJBROLE classes can be used to authorize WebSphere Application Server users to J2EE roles. If LDAP is used, the J2EE application will map J2EE security roles to RACF groups, which will contain the list of authorized users.

## 3.7 Authentication

*Authentication* is the ability to verify the users’ identity. The most common way to do this is by asking for something a user has and for something the user knows.

### 3.7.1 User IDs

Each human or non-human task must be personally identified as an authorized system user via a user ID. Each user ID is the individual’s unique identifier, which follows the task in the system for decision-making and auditing purposes.

RACF user IDs can be from one to eight alphanumeric characters in length. TSO user IDs can be up to seven characters. Non-TSO (including batch, system process, CICS, and UNIX® System Services) users can use up to eight character user IDs. RACF user IDs can contain components called segments, which extend the user ID’s reach into the environment. For example, users

needing CICS access might require a CICS segment, while UNIX System Services users might require an OMVS segment. All users accessing WebSphere Application Server applications require an OMVS segment, either specifically with their own RACF user ID OMVS segment, or via the default OMVS segment defined in RACF profile BPX.DEFAULT.USER in the FACILITY resource class.

### 3.7.2 Passwords

Passwords are used to verify something the user personally knows which authenticates that they are the authorized task for the user ID. Passwords can be up to eight characters in length and might have requirements for a particular pattern mask (for example, the password should not be the same as the user ID or the company name).

Other password requirements could also be implemented:

- ▶ Prevent reuse of previous passwords up to the last xx passwords.
- ▶ Special character syntax (for example, number in the third position, alphanumeric in positions 1-7).
- ▶ Passwords must be changed within a certain number of days.
- ▶ Passwords must not be changed within a certain number of days.
- ▶ Allow both upper-case or lower-case passwords.

### 3.7.3 Reauthentication

In this section we mention some techniques and considerations for reauthentication.

#### **RACO (ENVR, Environment Object)**

Since an Access Control Environment Element (ACEE) exists only within an address space, it cannot be used if a transaction running under a user's identity is passed off to another application contained in another address space. So how does RACF pass credentials to another address space? This is the purpose of the Environment Object (otherwise referred to as an ENVR or RACO). RACF can pass a user's credentials from one address space to another by the use of an ENVR, which is a transportable form of an ACEE and can be passed from one address space to another. When the new address space receives the ENVR, RACF can create a new ACEE for that user that is valid in the new address space. This ability to pass RACF credentials ensures that when a transaction, running under a user's identity, calls another WebSphere address space, the credentials under which the transaction started are passed to the new address space, and the transaction continues to run under the same identity.

## **SSO**

Any single sign-on application should revalidate users upon return to a connected application. “Spoofing” should not be permitted, and the integrity of the user access should be maintained.

## **Accessibility**

A user should be able to re-enter the system upon revalidation and access all resources they normally use.

## **Corporate requirements**

There might be corporate requirements guiding the installation on authorized use and methods for reauthentication of system users.

# **3.8 Authorization**

*Authorization* is the ability to determine whether an authenticated user can access a particular resource. The security product (for example, z/OS Security Server (RACF)) contains security rules that identify the resource being protected, a list of pre-approved user IDs with a specific level of access, and a default level of access for universal users. The security product check is invoked from a request of a resource manager (for example, DB2 or LDAP).

## **3.8.1 Role-based security design**

Companies should try to administer their security using logical groupings. J2EE security uses the concept of roles to assign common accesses to a group of people with the same access needs. Good design combines common J2EE role definitions into a grouping paradigm that allows for authorizing users to the group of roles without extra administrative overhead. RACF EJBROLE profiles can be used if “Local OS” is the defined user registry in WebSphere Application Server.

## **RACF groups**

Most users of a system need access to specific data or need the ability to administer a subset of users or resources. RACF groups serve an important role in administering the needs of a large user base. Groups allow a security administrator to gather together RACF user profiles that need the same level of access. Instead of giving users the authorities that each one needs, you create a group structure that mimics a company’s infrastructure and assign users to the correct groups that represent their needs. All RACF user profiles must belong to at least one RACF group but can be part of a number of groups. The group structure is implemented in a hierarchical tree structure. Each group on the tree

has either a child or parent group or both. A parent group can be thought of as a superior group. The purpose of a group hierarchy is only for group management. A superior group is the owner/administrator of the subgroup, but there is not any inheritance of privileges. Within the scope of WebSphere, groups may control access to application servers, back-end data, methods within applications, and many other areas. RACF groups can be named from one to eight alphanumeric characters and contain users who are “connected” with common resource access requirements.

### **Naming conventions for roles**

Roles should be given meaningful names that are flexible and easy to administer. The security administrator and WebSphere Application Server application developer should work together to develop good naming conventions.

## **3.8.2 RACF resources**

When a program accesses an object that is protected at the operating system level, an authorization check is performed by the security manager (for example, RACF.) Typical z/OS objects that would be accessed are z/OS data sets, UNIX System Services file system objects, connections, and TCP/IP ports. In WebSphere Application Server, ACEEs are associated to the region’s STCs and their address space. A profile can be defined for all resources, and these profiles can control access to these resources. Resources are controlled by a resource manager; these managers are responsible for calls (using SAF) for authorization checks. These resources also have related classes, which must be activated. When a resource class has been activated, the resource manager will recognize that it is active through return codes from access control validation and start learning the correct security information.

## **3.9 Auditability**

*Auditability* pertains to the creating and monitoring of system activity by users. This can include system access, resource access, and changes to system setting parameters.

### **3.9.1 Corporate requirements**

Many companies have increased audit requirements due to new government laws. For example, the United States’ Sarbanes-Oxley Act of 2002 (HR 3763) requires new audit and reporting mechanisms for financial and accounting firms.



With any security program, it is important to be able to perform an audit of accesses and violations of subjects. These audits are for both security reasons and change control processes. RACF has the ability to audit and report what resources are being accessed and by whom by creating SMF data records. Auditing tools can take this SMF data and create reports that can be analyzed and processed. Auditing is important because accountability requirements make it necessary for user credentials to be used from the initial system access and traceable entirely to the back-end system and data access.

### **3.9.2 Carrying forward security credentials**

A users' system activities should be traceable on request. System processes have to track and record activity by user IDs against system resources being accessed by the user IDs. A set of audit trails must be available for real-time system monitoring and later review.

An active user credential should carry forward in z/OS as the user pursues accessing system resources. These credentials should not change and should be non-repudiated where there is no doubt that the user accessed the resources at the specified date and time.

Propagation of a user ID is the ability to pass an authenticated user ID to one or more application components in a way the application can use. Propagation of security credentials is the ability to pass a user's authorization from one application component to another so that each application can make appropriate security decisions to access resources.



## A sample solution

In this chapter we go more into detail about the “security gates” securing access on the way to backend access on z/OS. Our main focus is the bridged scenario as described in 2.7, “Bridged security between z/OS and distributed” on page 20. It includes parts of all scenarios described in Chapter 2, “End-to-end security scenarios” on page 13. The bridged-security scenario might be similar to what many companies could consider to be the best solution for incorporating Web-based workload accessing their mainframe environments without feeling that they are compromising their historically tightly secured access to backend subsystems. The first concern in many z/OS environments will be how the deployment of Web-based workload onto their mainframes will affect security, and specifically the position of the z/OS Security Server (RACF) in the whole picture. This chapter will attempt to elaborate on some of the scenarios and show how RACF is integrated in the process.

We also discuss the SDBM LDAP back-end scenario, because it offers interesting possibilities for large intranet solutions, and its implementation in a WebSphere environment has not been widely described elsewhere.

## 4.1 LDAP SDBM registry

SDBM provides native authentication on z/OS with RACF without the need to configure DB2, or even install DB2 if you do not have it, as you would have to do for implementing a TDBM backend. The only prerequisite for using LDAP with an SDBM backend is RACF. The drawback might be that no other information than standard RACF fields is available on a common name. The functionality of LDAP with an SDBM backend could be seen as a pass-on to RACF. As mentioned before, setting up LDAP SDBM is simple and straightforward. J2EE role mapping is, when WebSphere is configured to use LDAP SDBM as user registry, a mapping of RACF users or groups to J2EE roles instead of a mapping of LDAP DN to them. This includes the administrative console roles Administrator, Configurator, Operator, and Monitor.

### 4.1.1 Setting up LDAP with an SDBM backend

LDAP is configured as an SDBM backend as follows. Example 4-1 shows the LDAP configuration file `slapd.conf` configured for SDBM.

*Example 4-1 slapd.conf entries for SDBM*

---

```
listen ldap://:3399
maxConnections 2000
adminDN "profiletype=user"
database sdbm GLDSDBM
suffix "sysplex=its,o=Itsopok"
sizeLimit 2000
timeLimit 3600
```

---

Refer to *z/OS Security Server LDAP Server Administration and Use*, SC24-5923, for a complete description of keywords.

Notice how only `profiletype=user` is specified in the configuration file. This enables multiple Bind Distinguished Names (BDN) to act as an LDAP AdminDN. The Bind Distinguished Name must be specified in the LDAP user registry properties under Global security → User registries in the administrative console. All security requests from WebSphere will be transferred to RACF (after Global security has been activated) under the LDAP Administrator DN of the Bind Distinguished Name. The BDN must be an RACF-defined user with a valid OMVS segment. This RACF user ID must have the system-wide AUDITOR attribute. An equivalent authority in RACF could be established by implementing an RACF command exit at dynamic exitpoint IRREVX01. An implementation of a command exit has the benefit of giving the possibility to restrict a user from alter commands that can be issued when the systemwide auditor attribute is assigned

(for example, the setting of audit options, refreshing classes) by limiting the user to list-only commands (LISTUSER, LISTGROUP, SEARCH, and so on).

Choose an appropriate WLM serviceclass for the LDAP server STC. SYSSTC is a good and valid choice, because most of the processing involved in authentication or authorization calls will be done in the WebSphere and RACF subsystems, not in the LDAP STC itself. SYSSTC will guarantee as much as possible that LDAP will not be a bottleneck, and assigning SYSSTC to LDAP will not put a heavy load on the system.

### 4.1.2 WebSphere user registry settings

In WebSphere, some Global security settings have to be set in order to use LDAP SDBM as user registry.

In the administrative console go to **Security** → **Global security** → **User registries** → **LDAP** and supply the values shown in Table 4-1.

Table 4-1 User registry settings

Property	Value (description, actual)
Server user ID	Master Administrators' RACF user ID
Server user password	Master Administrators' password
Type	Custom
Host	IP address or URL of LPAR where LDAP is listening
Port	LDAP listen port as specified in slapd.conf
Base distinguished name (DN)	suffix as in slapd.conf (without the quotes)
Bind distinguished name (DN)	racfid=BDNracid,profiletype=user,suffix
Bind password	password of BDNracid

**Note:** *BDNracid* is the BDNs RACF user ID and *suffix* is the string specified in slapd.conf: sysplex=its,o=Itsopok

The “Ignore case for authorization” check box should be checked in order to convert any lowercase user ID or password forwarded to RACF to uppercase. The decision whether to check the SSL Enabled checkbox depends on where the WebSphere cell you are configuring is located. If the WebSphere Application Server cell is residing on the same LPAR as the LDAP server, encrypting transport between the LDAP and WebSphere Application Server might be less

important than for a WebSphere Application Server distributed using RACF via LDAP SDBM for security purposes.

**Note:** SAF z/OS v1r7 (RACF release HRF7720) supports mixed-case passwords. The SETROPTS MIXEDCASE/NOMIXEDCASE parameters on the PASSWORD option determine whether the system allows mixed-case passwords. For more information, refer to *z/OS V1R7.0 Security Server RACF Security Administrator's Guide*, SA22-7683.

In Advanced Lightweight Directory Access Protocol (LDAP) user registry settings, add the filters shown in Table 4-2.

Table 4-2 Filters for Advanced LDAP user registry settings

property	value
User filter	racfid=%v
Group filter	racfid=%v
User ID map	*:racfid
Group ID map	*:racfid
Group member ID map	racfconnectgroupname:racfgroupuserids

As mentioned before, the Bind Distinguished Name should be an RACF user ID with the AUDITOR attribute, a valid OMVS segment (specific or implied by a defaulted segment), and *not* have a TSO segment. It does not need it, so it is an easy step to avoid misuse of the BDN account. Additionally, we recommend a non-expiring password for the BDN user ID. This avoids unpleasant surprises when the expiry date has come and the WebSphere cell comes to a halt due to internal authentication and authorization failures and failing user sign-on to applications. If supplying this category of user IDs with a non-expiring password does not comply with corporate policies, be sure to have processes defined and in place to change the BDNs password before expiry date, and plan a recycle of the entire WebSphere cell.

Distributed WebSphere environments can be configured in a similar way to use the LDAP SDBM server on z/OS as user registry.

### 4.1.3 Security flow

The security flow follows the diagram as pictured in Figure 2-2 on page 15:

1. An HTTP request is sent by a user. The HTTP server receives the request. Transport between requestor and HTTP server should be encrypted.

Optionally, HTTPS can be enforced by setting the Normal Mode directive to Off in the HTTP configuration file to exclude non-encrypted traffic. The HTTP server transfers the request to the WebSphere Application Server.

2. WebSphere Application Server sends back the login page (typically a form-based login). The user enters an ID and password. WebSphere Application Server authenticates through LDAP SDBM and builds an LTPA token. The user is authenticated and credentials are encrypted with the LTPA key in the token.
3. Access to parts of the application that are logically grouped under a security constraint by a J2EE role are checked against the credentials of the user.
4. Depending on one or more factors (the type of connector, application, or WebSphere settings), the user ID is propagated without change to an external security interface (EIS) or another identity can be assigned using techniques such as JAAS login modules, runas and Sync-to-OS Thread. All applicable RACF classes will be checked in the standard SAF manner that secure access to a particular resource.

**Note:** J2EE 1.3 specs for the form-based login have several limitations: expired passwords are not handled, the error-page is static, and it does not supply enough status information. Enhanced form-based login is an IBM extension to the J2EE 1.4 spec (WebSphere 6.0.x), which solves those problems. WebSphere 5.x users have to code their own extensions to the form-based login.

The described security flow is a high-level overview of the steps involved in a user request starting in a browser session, ending up in an EIS system and returning results to the browser session. It is also one of many possible configurations.

#### 4.1.4 Benefits of an LDAP SDBM solution

An LDAP server set up to run with an SDBM backend offers several benefits to an organization. The simplicity of setting up both LDAP and WebSphere for using RACF through SDBM as a user registry, and the possibility connecting distributed WebSphere environments to that same user registry gives an enterprise an ideal environment for testing purposes. Because all users have to be defined as RACF users on z/OS, application development can test their applications on a distributed WebSphere Application Server using authentication and authorization to resources as if they were running on a z/OS-located WebSphere Application Server. Also, connectivity to back-end EIS and all security-related issues involving that access to EIS can be tested, without having to re-implement the security infrastructure around their applications.

Another benefit of this setup is that there is no need to synchronize user registries, because all administrative security work is centralized on z/OS.

## 4.2 Security integration between z/OS and distributed

In this chapter we elaborate on the bridged security solution that we outlined in 2.7, “Bridged security between z/OS and distributed” on page 20. The issues an enterprise faces when planning for a Web-based application infrastructure include: What is the background for the scenario? As a result of z/OS and SAF design, all access to any resource on the system must be done by a valid SAF identity, and the possibility of creating audit trails either on accessed resources or the identity exists. The biggest issue in including WebSphere Application Server on z/OS in end-to-end security flows over existing platforms and existing security infrastructures is how to integrate those different user registries.

In a developing world and with emerging SOA architecture technologies, mainframes have to be flexible and able to integrate with the distributed world. At the same time, the mainframe needs to keep its main advantages, such as executing services with the best possible quality of service, workload scheduling, and so forth. In an SOA architecture exploiting distributed applications and services, security (read authentication and primal authorization) cannot be managed for Windows platforms on one side, Unix platforms on the other and the mainframe platform on yet another side. The end-user should be known on all platforms when a service is requested that might travel over different platforms. The obvious and most likely the simplest way to achieve this is to implement an enterprise-wide user registry.

When integrating WebSphere Application Server on z/OS into an enterprise-wide user registry, would we lose our long-established SAF security infrastructure on our mainframe systems? Absolutely not! A user registry is what it is: a registry containing attributes for all users that it manages. All low-level security outside of the user registry related WebSphere scope is SAF. WebSphere Application Server on z/OS can benefit from participating in that enterprise-wide user registry through manageability of user identities company-wide, by not having to synchronize user registries, and by not needing to use “glue” to integrate WebSphere Application Server on z/OS in an enterprise-wide Web transaction for the sake of maintaining an SSO implementation. It is important to realize that in order to integrate WebSphere Application Server on z/OS with the distributed environment in a user registry scope or other, the level of distributed security



must be at a high level, trying to match up with mainframe security standards. So how do we fit in WebSphere Application Server on z/OS in an enterprise-wide user registry? Here are some possibilities:

- ▶ IBM Directory Integrator (IDI or Tivoli DI): Build a virtual central directory based on existing directories aided by user registry synchronization products.
- ▶ LDAP on z/OS: Make LDAP on z/OS the central user directory, centralizing user management on the mainframe. The LDAP registry could be the user registry for Tivoli Access Manager. LDAP could be set up to use either an SDBM or TDBM backend, depending on the amount of LDAP attributes you want to supply on a Distinguished Name, in order to integrate with RACF.
- ▶ Tivoli Identity Manager: Use in combination with WebSphere Portal Server, Tivoli Access Manager, and Tivoli Directory Integrator for full integration of user registries, authentication, and authorization centralization.

The list can be extended; there are more options that we do not cover in this paper. The key message is that the possibility to use an enterprise-wide user registry is a reality today, and that it offers the most logic and clear implementation of a cross-platform end-to-end security integration.

#### 4.2.1 Bridged security solution

Setting up an environment to meet all possibilities as previously discussed is not accomplished quickly or easily. It involves a lot of planning of the architectural overview of an integrated security setup, and a lot of communication between the different platforms' security administrators. Also, integrating WebSphere Application Server on z/OS into enterprise-wide Web transactions might not be the first concern of an enterprise starting a WebSphere Application Server on z/OS project. We need to provide a feasible and sufficient secure solution in order to adhere to company requirements, and offer a secure alternative to two widely used, less secure ways of integrating WebSphere Application Server on z/OS in a distributed environment:

- ▶ Hard coding user ID and password in an application
- ▶ Keeping user ID and password stored in a properties (or other) file on distributed

The proposed scenario might not be the ideal solution from an integral security point of view, nor be a final implementation of accessing the backend connecting EJBs on WebSphere Application Server on z/OS from a manageability point of view. However, it offers a much more secure alternative to the two previously mentioned quick methods, and it does not involve a large transition road of synchronizing user registries or creating a new company-wide user registry.

## Distributed environment: the big picture

This chapter describes the front-end layout. This overview is not intended to be a precise guide to setting up or configuring all involved components but will point you in the right direction by describing the steps to take and by supplying links to documentation for more information.

At the heart of the front-end environment is WebSphere Application Server on distributed. WebSphere Application Server distributed is configured to use LDAP as the user registry; the same user registry is used by Tivoli Access Manager. Authentication and URI authorization (junction authorization for incoming HTTP requests) are handled by WebSEAL. WebSEAL, the Remote Proxy Security Server (RPSS) component of Tivoli Access Manager, is a Web server that listens on an HTTP port. WebSphere Application Server establishes a trust relationship with the RPSS using the Trust Association Interceptors (TAI) of the proxy server. At ID/password validation, a WebSEAL plug-in adds data from the reserved userdata field to the credentials of the user before the credentials are encrypted with the LTPA key. The custom data that is added to the credentials can be based on any relevant information, such as the users' Internet location or information retrieved from the LDAP user registry (as long as the diversity of the custom data is not as large as the amount of defined users in the user registry), and as long as the custom-built login module on the WebSphere Application Server for z/OS side can recognize it in order to map the supplied information to a system user.

For more information about adding custom data to credentials, refer to:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc/am60\\_web\\_devref.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc/am60_web_devref.pdf)

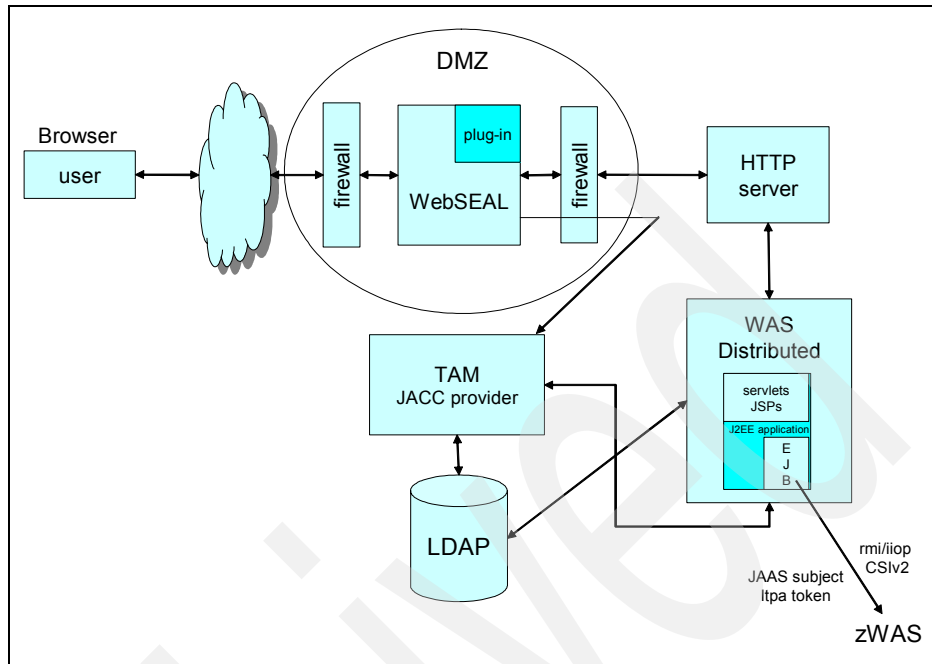


Figure 4-1 Distributed: big picture

**Note:** Strictly taken, the SSO domain in our scenario does not extend to WebSphere Application Server on z/OS, because WebSphere Application Server on z/OS (zWAS in the illustration) does not share the same user registry as the one used by WebSphere Application Server distributed (dWAS) and the Tivoli Access Manager cluster. However, there must be a trust relationship between WebSphere Application Server on z/OS and WebSphere Application Server distributed, because authentication of the enduser identity is not required at the z/OS side. This trust relationship in our scenario is established by using a shared LTPA encryption key on both sides, so effectively WebSphere Application Server on z/OS becomes part of the SSO domain because of not requiring re-authentication. The flow of an identity and a trust relation is referred to as *Identity Assertion*.

User credentials are forwarded to WebSphere Application Server distributed through an HTTP server serving the static content of the Web application. The HTTP server forwards dynamic requests to the WebSphere Application Server on distributed through the Web server plug-in. Tivoli Access Manager is configured in WebSphere Application Server distributed to serve as a JACC provider: J2EE role-based authorization decisions are made by Tivoli Access Manager based on policies. The Web modules of the J2EE application are deployed in the

WebSphere Application Server distributed. User credentials are checked to determine whether the user has access to the methods constrained by a J2EE role that will call the remote EJB on WebSphere Application Server on z/OS. Here we reach the point where propagation of user credentials, encrypted with the LTPA key into the LTPA token and containing the custom data in the userdata field, are propagated to WebSphere Application Server on z/OS over RMI-IIOP.

### **z/OS environment: the big picture**

Now that we have arrived in WebSphere Application Server on z/OS with the request, it is time to look at the component that does the mapping of the authenticated user, grouped to a more general identity than his user ID by the custom tag added at authentication in WebSEAL to the RACF system user.

### ***JAAS login module***

The key component in our scenario is a custom-built RMI\_INBOUND login module. The user identity is forwarded from WebSphere Application Server distributed to WebSphere Application Server on z/OS in the JAAS context. The login module extracts the custom data added to the user credentials and maps that data to an RACF user ID. A mapping table must be provided, either hard-coded inside the mapping module or somewhere in the filesystem. This RACF user ID is set as WSPincipal in the JAAS subject that will be used in the further thread. In this way we achieve a one-to-one or a more-to-one mapping of authenticated users on distributed to selected z/OS RACF user IDs. There are other ways to achieve more-to-one mappings that do not need custom coding of a login module, which we will explore later. Figure 4-2 on page 63 shows a graphical presentation of an RMI\_INBOUND flow. Three basic scenarios for an RMI\_INBOUND request can be described:

1. Already authenticated: an rmi EJB request and the CSlv2 session has already been established.
2. Already authenticated through a propagation login: If a token has been propagated from one server to another. All WebSphere Application Server login modules, as well as the custom login modules if provided, will be invoked.
3. Not authenticated: User has not authenticated. All login modules will be invoked as in the propagation login, but then in “initial login mode.”

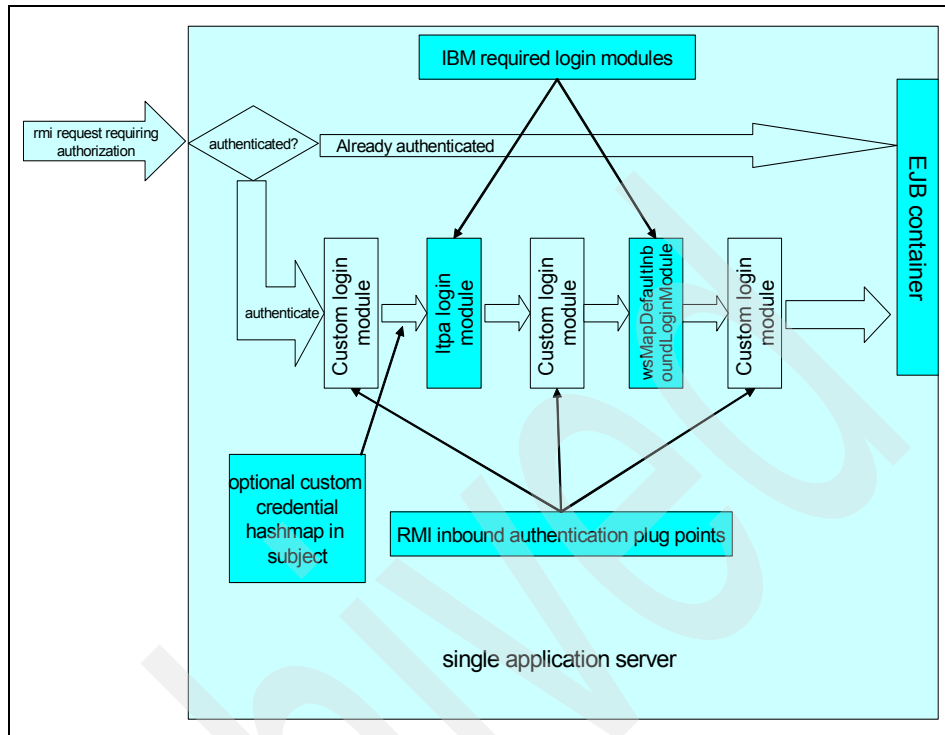


Figure 4-2 RMI\_INBOUND authentication flow

For more information about JAAS specifics, visit the Information Center, or read “IBM WebSphere Developer Technical Journal: Advanced authentication in WebSphere Application Server” at:

[http://www.ibm.com/developerworks/websphere/techjournal/0508\\_benantar/0508\\_benantar.html](http://www.ibm.com/developerworks/websphere/techjournal/0508_benantar/0508_benantar.html)

Because the user has already been authenticated and the LTPA token is sent in the JAAS subject over RMI-IIOP from the distributed environment, there is no question of an initial login. A CSlv2 session is being established at this point, so our login from WebSphere Application Server distributed fits RMI\_INBOUND scenario 2. For more information about writing custom login modules, visit this page of the information center:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rsec\\_jaascustlogmod.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rsec_jaascustlogmod.html)

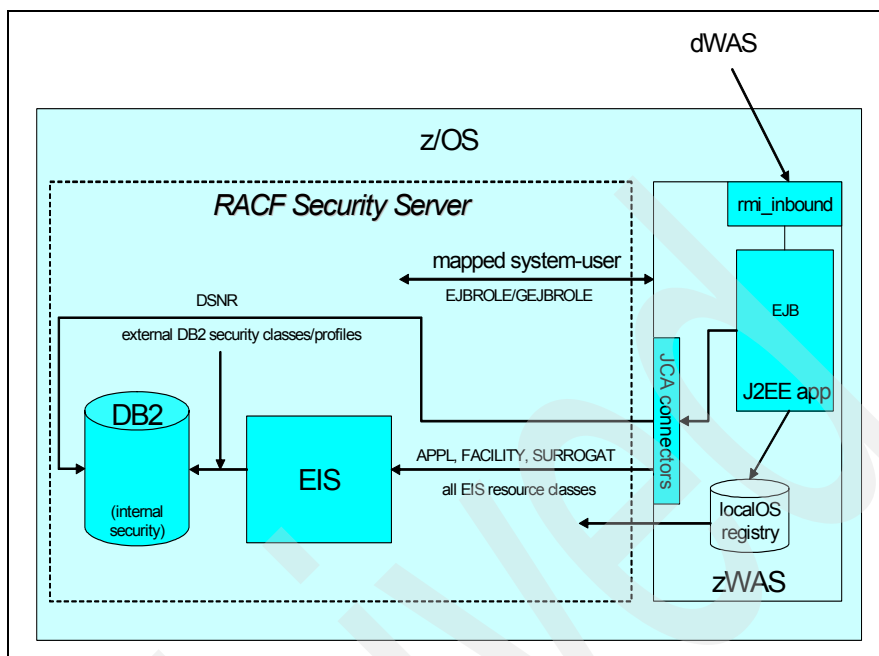


Figure 4-3 z/OS, the big picture

More information about general security concepts related to WebSphere Application Server 6.x can be found in many places. We recommend going through the “End-to-End Security Solutions for WebSphere on z/OS” presentation:

<http://wtscpok.itso.ibm.com/~wsweb/P0K00751/2005FDV2.pdf>

### ***EIS connectivity: Thread Identity, Thread Security, Connection Manager RunAs Identity, and Run-As Identity***

Now that we have set the Java principal to an RACF system-user that will be used to access the back-end EIS system, we go into detail about some additional WebSphere settings and J2EE terminology.

To access EIS as the J2EE identity that is either an authenticated user or a mapped identity by means of a login module (as is the case in our scenario), the OS thread must be set to run under this identity so it can access EIS systems with the credentials of the current Java principal. Set two options in the administrative console (**Security** → **Global security** → **z/OS security options**):

- ▶ **Support the synchronization of the OS thread**
- ▶ **Enable the connection manager RunAs thread identity**

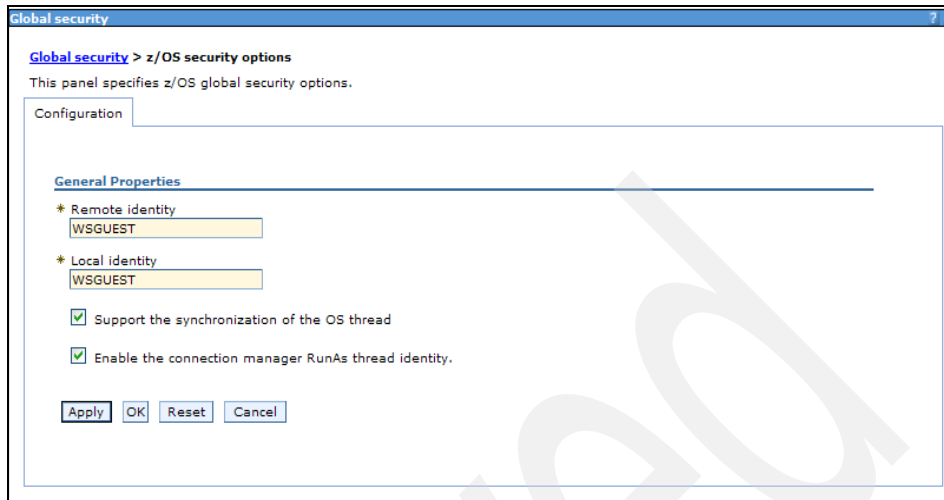


Figure 4-4 z/OS security options

### ***Sync To OS Thread, Thread Identity, and Thread Security***

The “Sync To Os Thread” thread security concept is unique to WebSphere Application Server on z/OS. It is used to indicate that when resources outside of WebSphere Application Server for z/OS are accessed, the user ID the process is running under is used. On other platforms this would always be the user ID that the EJB container is running under. That means that in the case of WebSphere Application Server on z/OS, the user ID under which the servant region STC is running would be used to access EIS or other resources on z/OS if the Sync To OS Thread option is set to *false*.

- ▶ *Thread Identity support* is the ability to pass the principal within the subject through a JCA connector to an EIS and is enabled only when:
  - A local connection is used between the application server and the EIS system (local to z/OS).
  - `res-auth=Container` is specified for the resource reference defined in the deployment descriptor of an application.
  - The connection factory does not specify a JAAS Authentication Alias.
  - You use a SAF-based user registry: localOS or LDAP native authentication.
- ▶ *Thread Security support* (also known as *Connection Manager RunAs Identity* or *Sync To OS Thread*) refers to the specific ability of WebSphere Application Server on z/OS to allow the switching of the servant regions’ TCB level ACEE

to the current J2EE identity. It applies only to JDBC™ access to DB2 or IMS and is enabled when:

- “Support the synchronization of the OS thread” and “Enable the connection manager RunAs thread identity” are set to true as shown in Figure 4-4 on page 65.
- A local connection is used between the application server and the EIS (for example, local to z/OS as opposed to a remote connection through a daemon).
- `res-auth=Container` is specified for the resource reference defined in the deployment descriptor.
- The connection factory does not specify a JAAS Authentication Alias,
- You use a SAF-based user registry: localOS or LDAP native authentication.

#### Notes:

- ▶ In fact, Sync To OS Thread applies when the application requires access to system resources (for example, HFS files, TCP/IP sockets). The application requests this function and the container will change the OS thread identity to the J2EE identity. Connection Manager RunAs identity applies to JDBC access to DB2 or IMS.
- ▶ JCA connection to CICS behaves differently because of the CICS Multi Region (MRO) design behind External CICS Interface EXCI connections via the CICS Transaction Gateway (CTG). Refer to the “Link security with MRO” chapter in the *CICS RACF Security Guide*, SC34-6011, for details about link-user and end-user resource accessibility.

A *JAAS Authentication Alias* is a reference to a J2EE Connector Architecture authentication data entry and is specified on a connection factory. The Authentication Alias is the user ID that is used to access the backend through this specific connection factory. This is one of the available mechanisms to have all connectivity to an EIS run under one predefined user ID (specified on the Authentication data entry).

*RunAs Identity* is the ability to change identity within an EJB to one of the following modes:

<b>RunAs Caller</b>	Use the identity of the caller.
<b>RunAs Server</b>	Use the identity of the container (servant region user).
<b>RunAs Role</b>	Use the identity assigned to an J2EE role.

RunAs Identity does not change the z/OS thread identity but is managed by WebSphere Application Server internally by changing the Java principal. However, when RunAs Role is specified in the deployment descriptor on a method



of a servlet that calls an EJB that accesses EIS, the user ID assigned to that role will be used to access the EIS. For example (Figure 4-5), if method AddUserAccount in servlet A specifies RunAs role(manager) and invokes an EJB that connects to an EIS, then the EJB will run under the J2EE identity of the user ID assigned to the EJBrole “manager.” Assigning a user ID to an EJBrole is done either by WebSphere Application Server bindings or in the EJBROLE SAF class on the APPLDATA field of the manager profile. Of course, the end user invoking the servlet method needs to have READ permission to the manager profile in the EJBROLE class in order to be allowed to invoke the EJB (isUserInRole). This Run-As scenario is another way to access back-end systems with a predefined user ID. It is much more flexible than assigning a JAAS Authentication Alias on a connection factory because its scope is on an application boundary instead of a connection factory boundary.

It is also possible to specify the RunAs attribute on the called EJB instead of specifying it on the calling method.

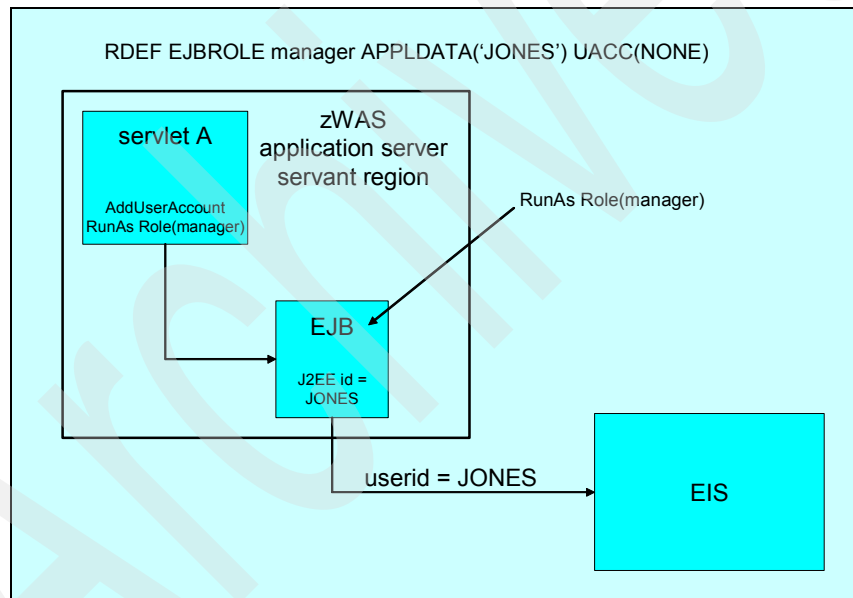


Figure 4-5 Run-As Role

**Note:** RunAs Server is an IBM extension to the J2EE specs.

For the bridged security scenario that we describe in this chapter we need Thread Security enabled, but we do not need either of the two mentioned ways to assign a fixed user ID to access EIS. We have established a more-to-one mapping to one or several RACF-defined identities. Because the principle in the

JAAS subject is that RACF identity, the RunAs Identity mode should be Caller, so the J2EE identity matches the previous principal and will be used to access EIS.

### **Alternative to the login module approach**

As an alternative to the method we described previously, which involves a plug-in on WebSEAL distributed side and a JAAS login module on z/OS side, it is worthwhile to mention the concept of *Certificate Name Filtering* in RACF. The end user logs on with a client certificate. The infrastructure mechanism to forward the certificate from WebSphere Application Server on distributed to WebSphere Application Server on z/OS is CSiv2 identity assertion (CSiv2 Attribute layer). The trust relation between WebSphere Application Server on z/OS and WebSphere Application Server distributed is established either through authentication at the CSiv2 Message layer or through authentication at the CSiv2 Transport layer, or through a trusted network zone.

### ***RACF certificate name filtering***

RACF provides a process for administering a large number of users without storing all of their digital certificates. RACF can selectively map users to either a unique user ID or a common user ID with auditability. RACF can associate many certificates with one user ID based on rules concerning portions of subjects or issuer's distinguished name (DN) in the certificate.

Certificate name filtering cannot be used in protocols where an actual certificate or private key is required. The RACDCERT command and parameters are used to create and maintain digital certificates and desired configurations. Benefits of this approach might be limited by a loss of granularity in access control.

An installation can define a specific set of certificate mapping rules. RACF will determine the proper user ID to use with its client certificate during initialization checking. Existing digital certificates can be used as model for creating relationships. RACF can define certificate mapping rules via:

- ▶ One-to-one certificate to user ID association
- ▶ Certificate name filtering
- ▶ hostIdMappings certificate extension

Classes involved are DIGTNMAP and DIGTCRIT. Name Filter mappings can have a one-to-one or a more-to-one relationship between the certificate and the mapped RACF user ID.

In Example 4-2 on page 69, the RACF identity is determined by a matching client certificate. The user ID JONES is associated with the client certificate because a matching certificate exists.

*Example 4-2 One-to-one mapping*

---

```
RACDCERT ID(JONES) MAP IDNFILTER('OU=ISSW,O=IBM')  
          SDNFILTER('CN=John') WITHLABEL('John')
```

---

Example 4-3 shows how RACF maps certificates issued by an IBM ISSW CA to RACF identity ISSWUSR. Certificate Filtering extends Mapping so distinct certificates do not have to be defined to RACF for every individual user (a major administrative task).

*Example 4-3 More-to-one mapping 1*

---

```
RACDCERT ID(ISSWUSR) MAP IDNFILTER('OU=ISSW,O=IBM') WITHLABEL('ISSW')
```

---

In Example 4-4, a certificate issued by IBM and by any CA but ISSW will be mapped to identity IBMUSR. RACF will map the certificate to the identity that best matches the issuer or subject distinguished name.

*Example 4-4 More-to-one mapping 2*

---

```
RACDCERT ID(IBMUSR) MAP IDNFILTER('O=IBM') WITHLABEL('IBM')
```

---

## 4.2.2 Overview of the benefits of a bridged security solution

A summary of the benefits of a bridged security solution to integrate WebSphere Application Server on z/OS in a distributed complex of WebSphere Application Server and Tivoli Access Manager includes:

- ▶ No need to synchronize or centralize plural user registries, and as a consequence no need to define all enterprise users as an RACF user ID.
- ▶ Security administration does not need to integrate cross-platform, and existing security skills and knowledge are leveraged at their strength.
- ▶ The time involved to achieve the integration will be relatively short compared to a user registry centralized solution.
- ▶ Segregation of EIS security from the distributed world: business as usual.
- ▶ Besides serving the enterprise as part of a larger cross transaction model, the WebSphere Application Server on z/OS is available as a separate, stand-alone application server. Standard RACF user IDs can access the WebSphere Application Server on z/OS as an application server.

Possible disadvantages include:

- ▶ The need to code and maintain two custom-built exit points: the WebSEAL plug-in and the RMI\_INBOUND login module.

- The concern about an end-to-end audit trail. The z/OS thread will run under the assigned system user. According to customers' requirements and policies, including a unique WebSphere Application Server transaction ID or flowing the distributed user ID at the message level could be implemented programmatically.

**Note:** A new SAF feature is planned for z/OS Version 1 Release 8 called SAF Identity Token that will benefit end-to-end auditability. This is an excerpt from the z/OS 1.8 preview:

Support for SAF Identity Tokens is planned in z/OS v.1 release 8 to provide exploiters with increased user accountability and auditability of resources by providing end-to-end auditing that tracks identities used for initial authentication and those used on the current platform.

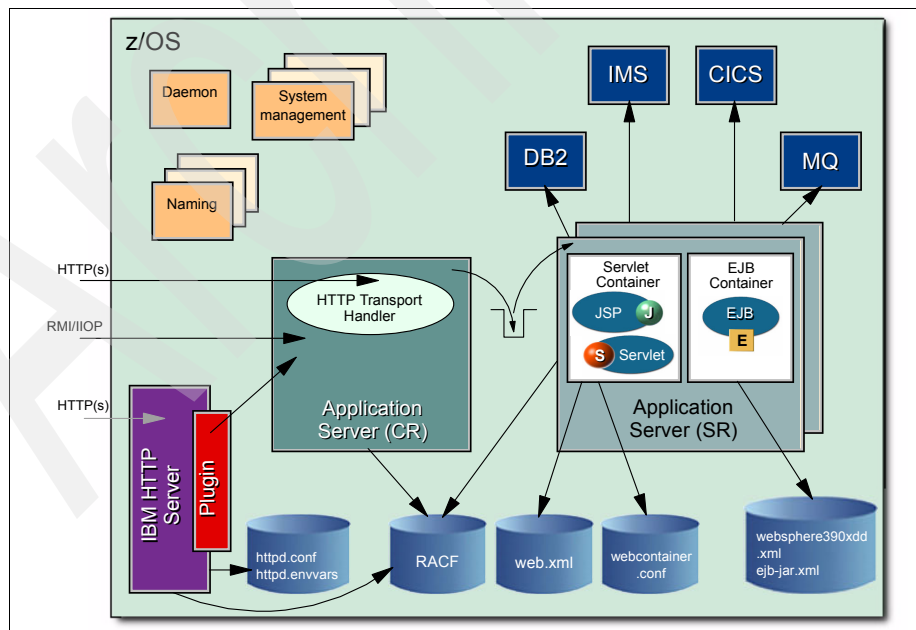
The z/OS 1.8 preview announcement letter is available at

<http://www.ibm.com/servers/eserver/zseries/zos/>

The versatility and flexibility of WebSphere Application Server on z/OS find its foundation in the infrastructure of z/OS. Scalable workloads, sysplex integration, quality of service (QOS), reliability, and the overall strength of z/OS security services make WebSphere and z/OS an unbeatable combination.

# J2EE security

This section provides a high-level overview of security features and functions for WebSphere Application Server for z/OS Version 6 and J2EE 1.4 security. This figure shows the runtime environment of WebSphere Application Server on z/OS.



## J2EE 1.4 security features

WebSphere Application Server implements a pluggable security model that you can configure to your current environment and existing IT resources. This architecture provides the following types of security.

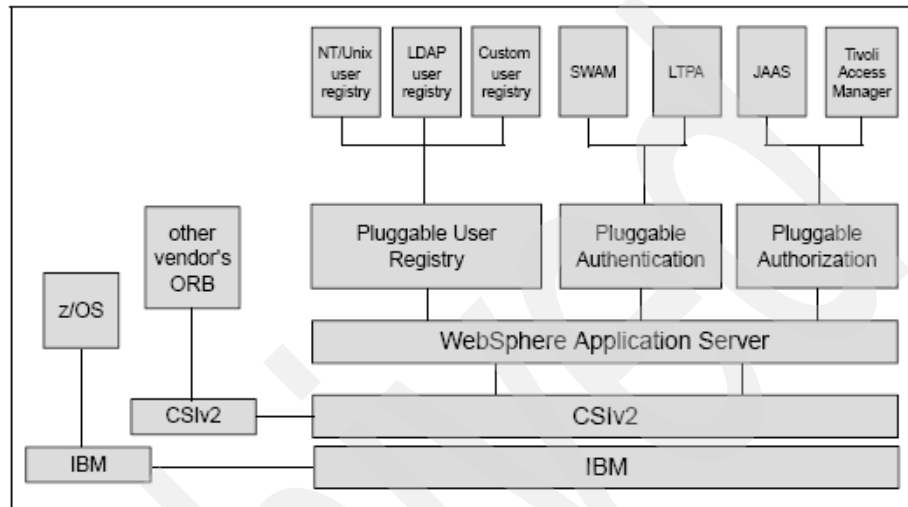


Figure A-1 WebSphere Application Server V6 security features

Most of the security functions and administration are similar in V6 and V5 of WebSphere Application Server. Something new in V6 is the support for Java Authorization Contract for Containers (JACC). In addition, WS-Security specification has been made more current.

J2EE 1.4 Security defines the following specifications, which are supported by WebSphere Application Server Version 6:

- ▶ Java 2 security
- ▶ JAAS security
- ▶ J2EE security roles
- ▶ CSIv2
- ▶ JACC

These specifications are distinguished by the way the security is enforced.

### Java 2 security

The purpose of Java 2 security is to prevent application code from gaining access to system resources such as files, ports, and sockets in a malicious or unintended manner.

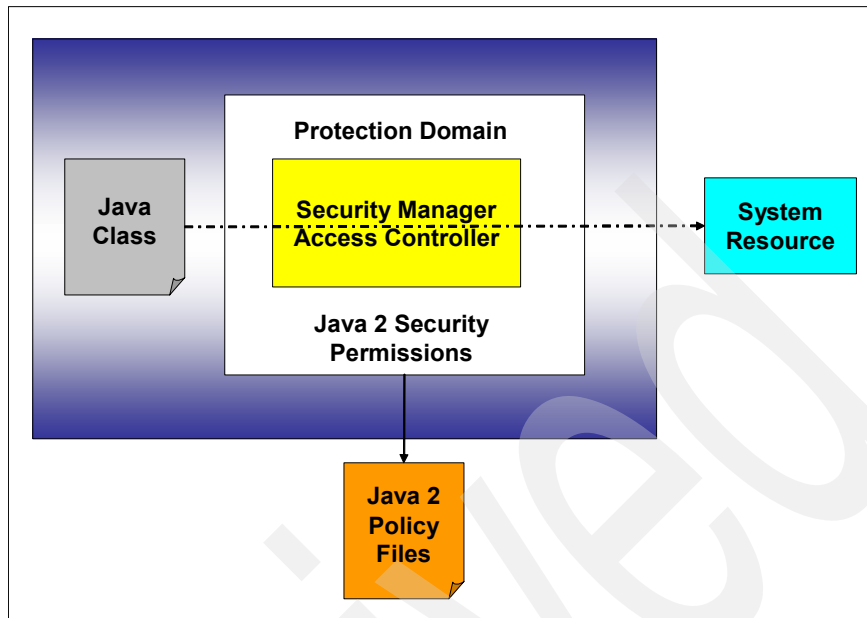


Figure A-2 Java 2 security

Before allowing code to access a system resource, the access control mechanism of the JVM™ will read the policy files to determine whether the correct permission is provided. Otherwise, no access will be given. If your application code must access system resources, you have to grant the necessary permission in policy files.

Java 2 security is optional and is enabled automatically when global security is turned on. You can configure Java 2 security and global security independent of one another. Disabling global security does not disable Java 2 security automatically. You need to explicitly disable it and restart your application server for the changes to take effect.

There is significant room for discussion about whether Java 2 security is a big benefit when WebSphere is on z/OS. Access to HFS/zFS files, ports, and sockets are typically (should be anyway) locked down by RACF. In general, Java 2 security is redundant in a secure environment such as z/OS. Many customers have saved the overhead (3-4%) of Java 2 security by disabling it. Now, there is some risk. If the application does a `system.exit()` it will cause the servant to recycle. (We directed the JVM to shut down.) Of course, server-side programmers should never do this and it is an easy problem to uncover and correct.

Example A-1 shows some examples of system resources protected by Java 2 security. You must explicitly grant permission to the code or the user executing the code in order to access these resources.

---

*Example: A-1 Permissions protected by Java 2 security*

---

File Access:

canRead(), FileInputStream(), RandomAccessFile(), isDirectory(),  
isFile(), length(), canWrite(), FileOutputStream(), mkdir(),  
renameTo(), createTempFile(), delete(), deleteOnExit()

Network Access:

send(), receive(), getLocalAddress(), getHostName(), getLocalHost(),  
getAllByName()

Java VM:

ClassLoader(), loadLibrary(), checkPermission(), checkLink(),  
checkExit()

Program Threads:

stop(), resume(), suspend(), interrupt(), setPriority(), setName(),  
setDaemon()

System Resources:

getPrintJob(), setProperty(), getProperty(), setDefault(), getFont(),  
getEventQueue()

Security Aspects:

getFields(), getMethods(), getConstructors(), setPublicKey(),  
addCertificate()

---

Example A-2 shows an example of the syntax used in the policy file.

---

*Example: A-2 Policy file syntax*

---

```
keystore "URL" "keystore-type"
grant signedby "signer" codebase <Codebase URL>
{permission "class_name", "target" "action";
permission "class_name", "target" "action";
...
};
```

---

In this syntax:

- ▶ keystore file is the public key used to identify the signer.
- ▶ signedby is the name of the signer.



- ▶ codebase grants permission for code located in URL.
- ▶ permission to access "class\_name" to allow performing "action" on the "target"

Java 2 security permissions are granted through the use of policy files and must adhere to a strict format. Any formatting errors in the policy file will prevent it being processed.

Permission can be granted by codebase, specifying the location of the Java code that will access the resources specified in the permissions or by the signer of the code.

Best practice is to edit the policy files with the Policy Tool to ensure that the strict format is adhered. A hand edit of the server.policy file could be error-prone and could prevent the server from starting.

*Example: A-3 Grant Java code in directory:/home/MyProgram I/O permission to read/write /tmp/log.txt file.*

---

```
grant codeBase "file:/home/MyProgram/"{
    permission java.io.FilePermission "/tmp/log.txt","read,write"; };
```

---

WebSphere Application Server supports a hierarchical definition of policy files at application and server level. Both static and dynamic policy files are supported. Static policy files provide default permissions, and dynamic policy files provide application permissions. Examples of policy files in WebSphere Application Server for z/OS V6:

- ▶ Static policies
  - java.policy
  - server.policy
  - client.policy
- ▶ Dynamic policies:
  - filter.policy
  - spi.policy
  - library.policy
  - app.policy
  - ra.xml
  - was.policy

Filters in Java 2 security ensure that application developers do not grant applications permission to do things that could be dangerous, such as exit the JVM or create their own Security Manager and implement their own security. The effective application policy becomes:

app.policy + was.policy + java.policy "-" filter.policy

Administrators need to understand the possible consequences of enabling Java 2 security. If your applications, or third-party libraries have not been properly tested, enabling Java 2 security can cause problems. You can identify these problems as Java 2 security `AccessControlExceptions` in the system log or trace files. If you are unsure about the Java 2 security readiness of your applications, disable Java 2 security initially to get your application installed and verify that it is working properly.

For more details, search on “Configuring Java 2 security policy files” in the information center.

## Java Authentication and Authorization Service (JAAS)

Java Authentication and Authorization (JAAS) is a standard and pluggable Java framework that enables applications to authenticate and enforce access controls upon users.

JAAS provides two types of interfaces:

- ▶ An application-level programming interface (API) for use by applications
- ▶ A service programming interface (SPI) for the providers of its functionality.

WebSphere Application Server supports the use of JAAS for login and customized authentication. WebSphere Application Server provides the following JAAS support:

- ▶ Well-defined interfaces for altering the user subject
- ▶ Enhanced Trusted Association Interceptor (TAI) support
- ▶ Explicit documentation for the WebSphere Application Server login process
- ▶ The ability to assert complete user credentials to WebSphere Application Server (including group information)
- ▶ Replication of subjects in a distributed environment

JAAS uses the concept of a subject to define a user. A subject is created at initial authentication time and includes principal and credential data. (This would be `WSPincipal` and `WSCredential` objects in WebSphere Application Server.)

`WSPincipal` is typically a Java principal that is used to define an entity in Java such as a user, organization, or login ID. The `WSCredential` defines security information for authorization, such as group memberships.

Core JAAS objects are:

- ▶ Subject
- ▶ LoginContext
- ▶ LoginModule

Authenticating a subject follows these steps:

1. An application instantiates a LoginContext.
2. The LoginContext consults a login configuration to load all of the login modules that are part of that configuration.
3. The application invokes the LoginContext login method.
4. The login method invokes the loaded login modules. Each login module attempts to authenticate the subject. Upon success, login modules associate relevant principals and credentials with a Subject object that represents the subject being authenticated.
5. The LoginContext returns the authentication status to the application.
6. Upon success, the application retrieves the subject from the LoginContext.

## J2EE security roles

Intuitively, J2EE security roles provide role-based security. The roles are defined in the J2EE EAR file, specifically in the application configuration setting such as deployment descriptors. It can be enforced by the runtime, programmatically, or both. The defined security roles apply to the entire application, including all of its modules.

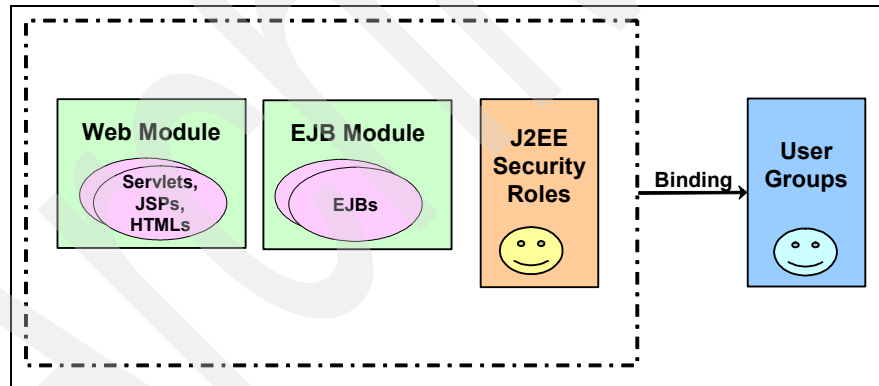


Figure A-3 J2EE security roles

*Binding* is the task of mapping the security roles to the users or groups and is normally performed by the system administrator when installing application. When z/OS SAF authorization is being used to check role membership, this entails having the security administrator give the required users access to the specified roles as defined in the EJBROLE profiles. With the support of JACC in V6 of WebSphere Application Server, you can use IBM Tivoli Access Manager or a third-party JACC provider to provide the binding information.

## Declarative security

*Declarative security* is enforced by the container, which removes much of the responsibility of security from the application developer. The declared security properties are defined in the J2EE components' deployment descriptors.

The J2EE specification defines declarative security as the means of expressing an application's security structure, including security roles, access control, and authentication requirements in a form external to the application.

## Deployment descriptor

The deployment descriptor is the primary vehicle for declarative security in the J2EE platform. It implements declarative security with definitions of security identities and security roles.

In the deployment descriptor of both Web applications and EJB applications, security constraints can be defined that have to be satisfied for the application to be allowed to run. These constraints can be defined on the level of individual methods or on the level of a logical grouping of methods.

## JAAS (Java Authentication and Authorization Service)

This service is used by WebSphere Application Server to:

- ▶ Identify the client
- ▶ Check client access to the WebSphere Application Server application
- ▶ Allow WebSphere Application Server usage by applications
- ▶ Create SSO LTPA tokens

## Permission-based model

Resources such as programs, files, or devices are associated with users or groups of users who are given various levels of permission with respect to those resources.

## Programmatic security

The security APIs used in the logic of application programs. Security role names are used in the APIs.

It is desirable to use declarative security wherever possible to relieve the application programmer of the task of having to add security-relevant code to an application. In cases where this is not feasible or where the program logic requires security to be handled by the program, programmatic security can be

used, as defined by the J2EE specification. Programmatic security refers to security decisions made by security-aware applications. Programmatic security is useful when declarative security alone is not sufficient to express the security model of the application.

## Roles

Security roles are a logical grouping of users who share a level of access permissions. During assembly of the J2EE application, permissions to call methods are given to various roles. Roles define a set of permissions in a J2EE application and are defined in a J2EE EAR file. During deployment, users or groups (or both) are assigned permissions to roles. Roles are associated to methods and are defined by the deployment descriptor. The role access is enforced at run time, programmatically, or both.

## RunAs

RunAs allows for one user to obtain the permission to use resources authorized for another user. A caller specifies to use the caller's identity for the method selected and to propagate it to any subsequent methods invoked or J2EE resources accessed. RunAs is used for "downstream" authorizations (the RunAs identity will be used as the Caller Identity in EJBs called by the current EJB), or "outbound" identities (the RunAs identity will be used as the user ID for EIS systems called from the current EJB when container-level authorization is specified). Figure A-4 on page 80 describes how RunAs processing uses the SAF EJBROLE class. Read more about RunAs in Chapter 3, "z/OS and WebSphere security technology overview" on page 25.

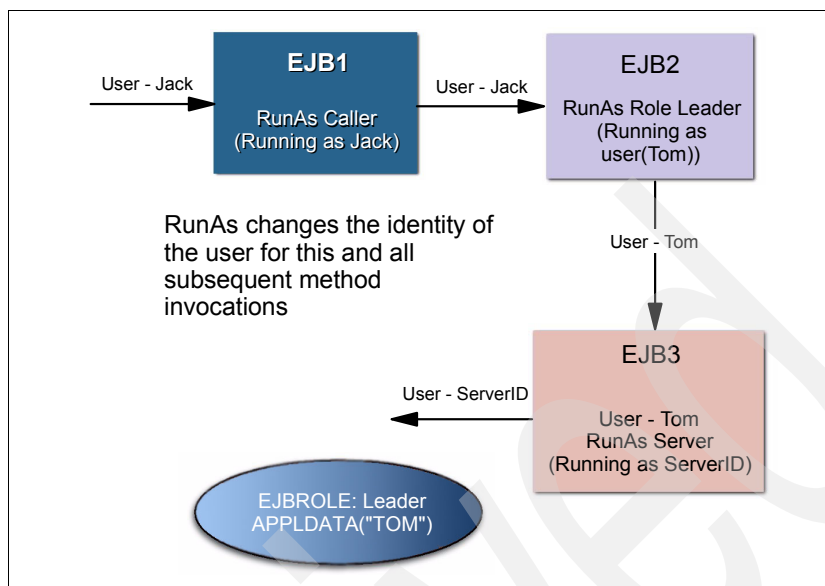


Figure A-4 RunAs and EJBROLE profiles

## CSlv2

WebSphere Application Server for z/OS V6 supports two authentication protocols:

- ▶ CSlv2 - recommended  
Defined by Object Management Group (OMG) and is part of the J2EE standards.
- ▶ z/OS Security Authentication Services (z/SAS) - for backward compatibility  
Used by previous levels of WebSphere Application Server.

In future releases, IBM will no longer ship or support the z/OS Secure Authentication Service (z/SAS) IIOP security protocol. It is suggested that you use the Common Secure Interoperability version 2 (CSlv2) protocols.

Common Secure Interoperability Specification, Version 2 (CSlv2) defines the Security Attribute Server (SAS) that enables interoperable authentication, delegation, and privileges.

CSlv2 is intended for use in environments where SSL and Transport Layer Security (TLS) are used at the transport layer to provide message protection and server-to-client authentication. CSlv2 configuration is integrated into WebSphere Application Server Security Administration.

An authentication protocol is required to determine the level of security and type of authentication that has to occur between the EJB client and the EJB for each request in a secure environment. An EJB client making secure RMI-IIOP calls to EJBs must provide authentication information. This is done by using either CSlv2 or zSAS.

Security features provided by CSlv2 include:

- ▶ **SSL client certificate authentication**

SSL certification authentication does not occur at the message level, but occurs during the connection handshake using SSL certificates.

  - Advantage of using a certificate is increased authentication security.
  - Disadvantage is the complexity of configuring each client with the proper keystore file.
- ▶ **Message layer authentication**

Message layer authentication uses a token to store and exchange credential information with the receiving server.

Tokens contain credentials in either the ID/password format or LTPA token.
- ▶ **Identity assertion**

Identity assertion uses a CSlv2 identity token to identify the client to the downstream without providing authentication data, which is helpful when there is no common authentication token. However, identity assertion requires a trusted channel between the asserting server and the one that receives the identity.
- ▶ **Security attribute propagation**

Secure attribute propagation enables authenticated subject contents and security context to be passed from one server to another.

  - Java objects contained in the subject must be serialized
  - The advantage is that downstream servers do not have to perform a lookup in a user registry in order to get attributes.
- ▶ **Stateful and stateless choices**
  - Stateful sessions require authentication only on the initial contact and therefore provide better performance.
  - For applications where authentication is needed for every contact, stateless option should be used.

This is a high-level authentication protocol flow:

1. Client ORB calls the connection interceptor to create the connection.
2. Client ORB calls the request interceptor to get client security information.

3. Server ORB calls the request interceptor to receive the security info, authenticate, and set the received credential.
4. Server ORB calls the request interceptor to allow security to send information back to the client along with the reply.
5. Client ORB calls the request interceptor to allow the client to clean up and set the session status as good or bad.

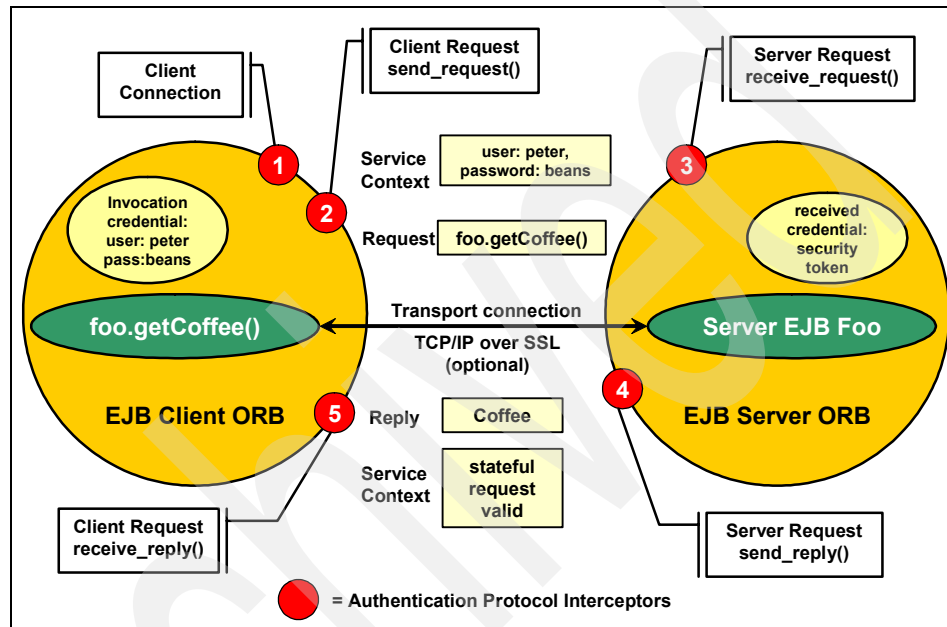


Figure A-5 High-level authentication protocol flow

### When to look into this?

When you are designing an application in which (RMI-IIOP) calls are being made to EJBs in WebSphere Application Server on z/OS. A call might come in from another server on the network and it might be trusted or untrusted. Unless you are 100% certain that all calls to EJBs on the System z server come from a trusted source, you should require authentication to take place; therefore, a security context is required as well.

## Java Authorization Contract for Containers (JACC)

JACC enables application servers to interact with third-party authorization providers using standard interfaces to make authorization decisions.



The security policy and user/group bindings to the security role are maintained by the JACC provider. JACC allows authorization information (J2EE Security roles to user/group binding) to be stored in external JACC providers.

JACC defines permission classes for both the EJB and Web containers. However, JACC does not specify how to assign principals to roles.

Support of JACC-based authorization providers is in addition to the support of the Default Authorization binding, using the IBM Binding file for authorization information or EJBROLE profiles if SAF authorization is used on z/OS.

Figure A-6 shows an example of the way WebSphere Application Server V6 interacts with a third-party JACC provider.

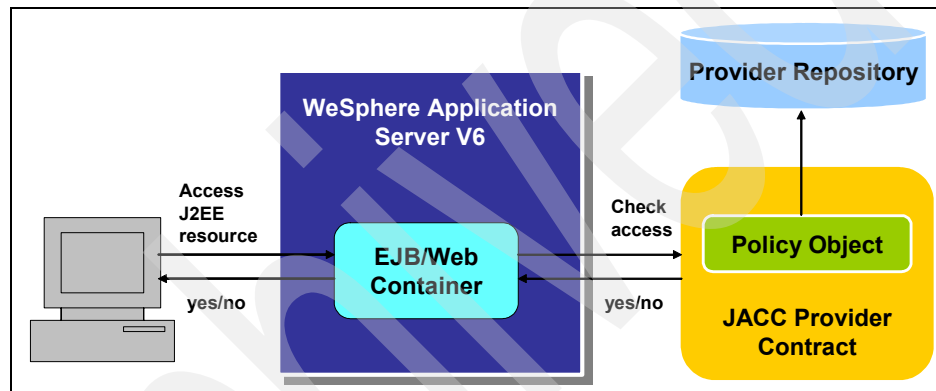


Figure A-6 JACC with WebSphere Application Server V6

1. Authenticate the user by checking the user's credentials.
2. Create a permission object for the resource being accessed.
3. Register required information by using the PolicyContextHandler objects.
4. Create the unique identity for the module being accessed.
5. Call the java.security.Policy object implemented by the provider to make the access decision.

When a module is being installed, the application server will generate the necessary permission objects with information from the deployment descriptors. This will be combined with a unique identifier for the application module and sent to the JACC provider. The JACC provider will store this information in its repository. During validation of the authorization permission process, the Application Server queries the JACC provider to get the roles associated with the user or group.

IBM uses IBM Tivoli Access Manager as a JACC provider.

# Putting together J2EE security

Figure A-7 represents how WebSphere Application Server for z/OS V6 incorporates J2EE 1.4 security features.

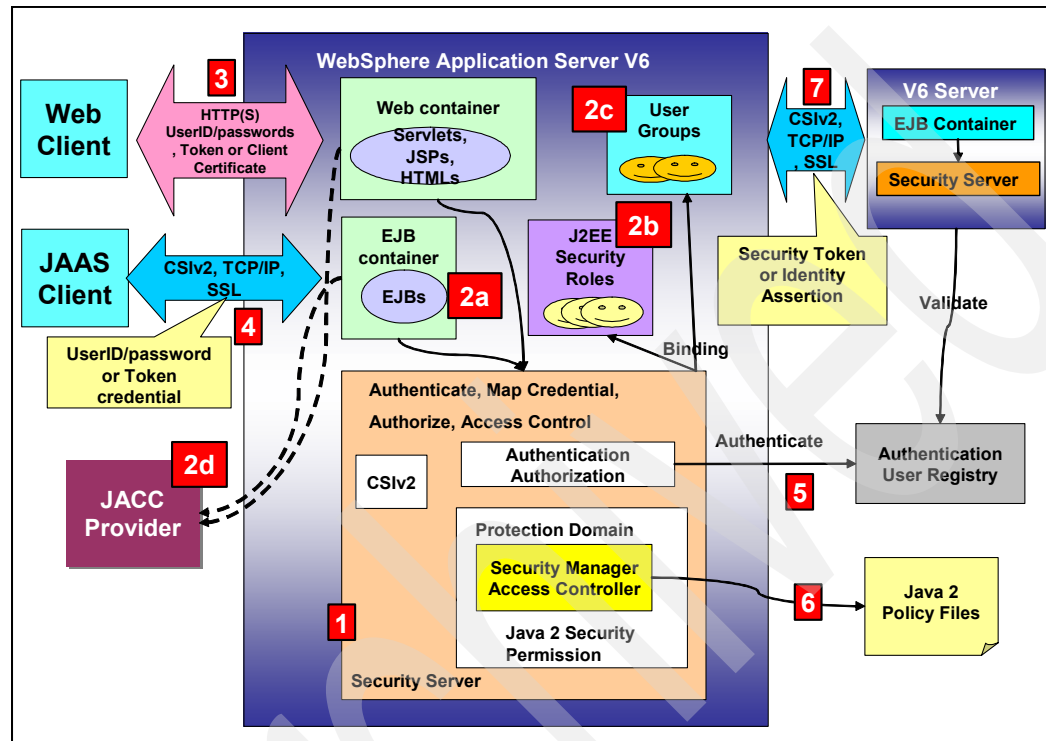


Figure A-7 J2EE security big picture

1. Security server in WebSphere Application Server v6 performs all security-related functions, such as authentication, authorization, checking for Java 2 permissions, and so forth.
2. Web and EJB applications (2a) with the J2EE security roles (2b) defined for the applications. J2EE security role to user/group binding (2c). This enables the security server to know which J2EE security roles the user/group belongs to. Alternatively, WebSphere Application Server can outsource the task of authorization decision-making process to an external JACC provider.
3. Web client requests directly or through a Web server using HTTP or HTTPS. When authenticated by the security server, authorization is checked based on J2EE security roles and permissions.

4. JAAS client calling EJBs using CSiv2 authentication protocol, making RMI-IIOP calls with or without SSL. When authenticated, authorization is checked based on J2EE security roles and permission.
5. All authentication is checked against the supported user registry. WebSphere Application Server for z/OS V6 supports the following user registries:
  - Local OS (system authorization facility)
  - LDAP
  - Custom user registry
  - Trust Association Interceptor (TAI), which uses Tivoli Access Manager for authentication.
6. Java 2 security policy files used by the JVM Access Controller to check whether the executing Java code has the necessary permissions to access the requested system resource.
7. Application in WebSphere acting as an EJB client to an EJB in another server and using CSiv2 authentication protocol.



## **z/OS Security Server (RACF)**

In this appendix, we provide an overview of the z/OS Security Server, or RACF. If you already feel comfortable with RACF, you might decide to skip this chapter.

## Technology

The design and development of the z/OS Security Server (RACF) follows a published architecture that adheres to industry standards. RACF does not modify or front-end any modules, nor does it dynamically place hooks into product code. RACF uses the z/OS-provided SAF interface. By following standards and using the SAF interface, RACF provides reliability and allows for ease of release-to-release migration.

RACF is designed to meet the performance demands of a highly active system. This is accomplished by many design features including the use of the Coupling Facility for buffering the database. There is an almost continuous process of adding new features and functions to meet the demands of business, including passtickets, one-time-use tokens that are used so that passwords do not appear over communication links and Digital Certificates to support user authentication across the Internet.

## Components

RACF is a single product that provides not only RACF, but the z/OS LDAP server, the z/OS firewall technologies, Network Authentication Privacy Service (Kerberos), z/OS Open Cryptographic Services Facility (OCSF), and the z/OS DCE server. Figure B-1 on page 89 provides an overview of z/OS security components.

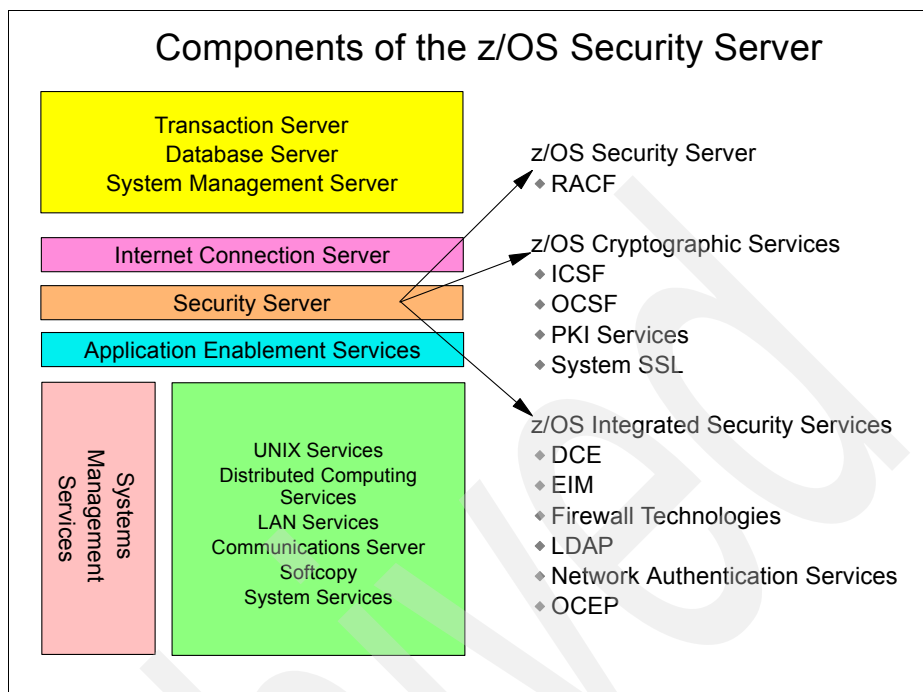


Figure B-1 Components of the z/OS Security Server

## z/OS Security Server (RACF)

This section describes the component in z/OS Security Server (RACF).

### RACF

The *Resource Access Control Facility (RACF)* is the only element of the z/OS Security Server. RACF implements and monitors the implementation of an installation's security policy on z/OS.

## Cryptographic Services

This section describes the components of the z/OS Cryptographic Services.

### ICSF

The *Integrated Cryptographic Services Facility (ICSF)* is designed for high security and high performance. It uses DES algorithms and maintains master keys in hardware. Cryptographic coprocessors and Trusted Key Entry workstations are supported. RACF provides callable services for applications to exploit use of encryption technologies.

## OCSF

The *Open Cryptographic Services Framework (OCSF)* contains a set of APIs for UNIX applications to use certificates and keys.

## PKI services

The *Public Key Infrastructure (PKI)* is a z/OS-based Certificate Authority that provides the complete Certificate Authority package for the z/OS environment.

## System SSL

SSL is a communications protocol for use by two applications communicating over an unsecured network. It is essential for secure transactions between a Web browser and a Web server and gives the option of using RACF digital certificate support.

## Integrated Security Services

This section provides an overview of the z/OS Integrated Security Services.

### DCE

The *Distributed Computing Environment (DCE)* is a self-contained environment and tools for developing and running applications on heterogeneous distributed systems.

### EIM

The Enterprise Identity Mapping (EIM) component supports multiple user registries and identities. It defines associations between an identifier and user IDs in registries that are part of operating system platforms, applications, and middleware.

## Firewall technologies

Firewalls are key network infrastructure components in security. They have many functions that help to separate network segments and provide services to connect them. A firewall is a “device” that is used to separate and allow selective access between a “safe” network and a “not-so-safe” network.

By separating network segments, communication can be controlled on the protocol level between clients and servers. A few security functions that firewalls can provide are:

- ▶ Hiding actual server names and addresses from outside connections
- ▶ Filtering communication based on originating addresses
- ▶ Filtering communication based on protocols
- ▶ Authenticating connecting clients



## LDAP server

The LDAP server provides the common user registry that can be exploited by WebSphere Application Server and other z/OS components. Figure 4-6 shows a high-level view of LDAP on z/OS.

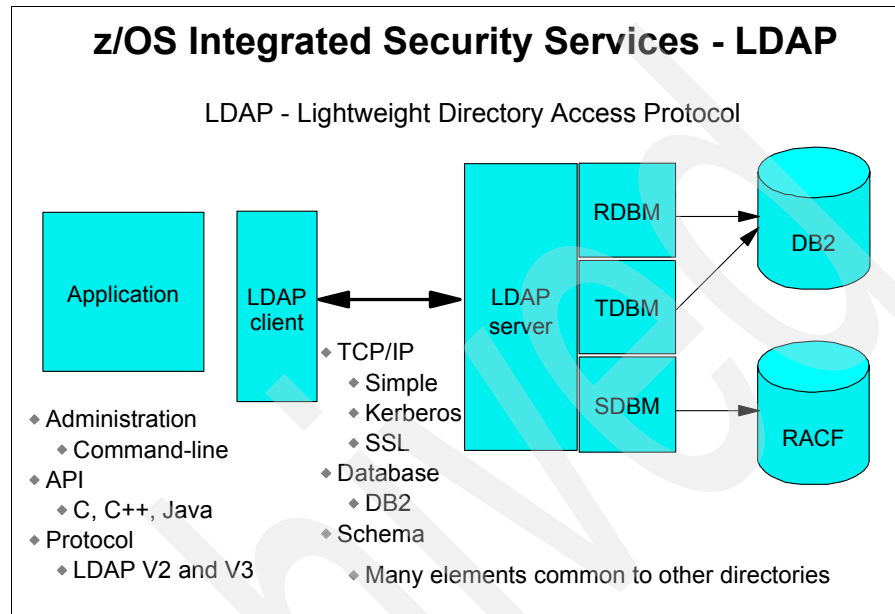


Figure 4-6 LDAP

## Network Authentication Services (Kerberos)

“Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.” (from the MIT Kerberos Web site). You can find this and more information about Kerberos at:

<http://web.mit.edu/kerberos/www/>

## OCEP

Open Cryptographic Enhanced Plug-Ins (OCEP) are an inventory of plug-ins that provide enhanced functionality for read-only access or to verify the trustworthiness of RACF key rings.

## Usability

In recent years, with expanded use of the Internet, open systems, and client/server systems, security has changed from just a product on the mainframe to include the complete enterprise. There is no single product that provides security for all of these environments, but there are products that provide management capability for all of these environments. RACF integration with and support by Tivoli provides for consistent security management from a single point or from distributed locations. Tivoli provides an open framework supported by many vendors of system management products. This provides the ability to manage diverse systems with consistency and integrity.

## Adherence to industry standards

z/OS Security Server (RACF) is consistent with the following industry standards:

- ▶ UNIX System Services: UNIX 98 branded, X/OPEN XPG4, POSIX 1003, and IBM Common Cryptographic Architecture (CCA)
- ▶ Cryptography: Intel® Common Data Security Architecture (CDSA), also known as OCSF on z/OS
- ▶ Digital Certificates: ITU-T X.509 certificates
- ▶ LDAP: ISO X.500 directories and registries, IETF standards
- ▶ Firewall: ICSA IPsec certified

## ACEE

If the user identification and verification are successful when a user logs into a system, that user is assigned a control block called an ACEE (Access Control Environment Element). This control block contains a description of the current user, including user ID, current connect group, user attributes, and group authorities and any credentials a user has. z/OS security forces all users who log into the z/OS system to get assigned an ACEE that follows that user around. The ACEE is a control block; therefore, it is valid only within the address space where it was created. In other words: an address space contains an application that a user wants to access. When the user accesses the application within that address space, RACF authenticates the user and, upon authentication, RACF creates an ACEE that represents that specific user and contains all of the user's credentials from RACF. Knowing this reveals some shortcomings that could be encountered when using LDAP as a user registry. Where is the ACEE created, and for what user, when the end user is represented in LDAP? Using the LocalOS user registry is a much easier story. WebSphere authenticates the user

and the ACEE is created. When authenticated to LDAP, no RACF call has been made and no ACEE creation is requested.

## **RACO (ENVR, Environment Object)**

Since an Access Control Environment Element (ACEE) exists only within an address space, it cannot be used if a transaction running under a user's identity is passed off to another application contained in another address space. So how does RACF pass credentials to another address space? This is the purpose of the Environment Object (otherwise referred to as an ENVR or RACO). RACF can pass a user's credentials from one address space to another by the use of an ENVR, which is a transportable form of an ACEE and can be passed from one address space to another. When the new address space receives the ENVR, RACF can create a new ACEE for that user that is valid in the new address space. This ability to pass RACF credentials ensures that when a transaction, running under a user's identity, calls another WebSphere address space, the credentials under which the transaction started are passed to the new address space, and the transaction continues to run under the same identity.

## **Groups**

Most users of a system need access to specific data or need the ability to administer a subset of users or resources. RACF groups serve an important role in administering the needs of a large user base. Groups enable a security administrator to gather particular RACF user profiles that need the same level of access. Instead of giving users the authorities that each one needs, you create a group structure that mimics a company's infrastructure and assign users to the correct groups that represent their needs. All RACF user profiles must belong to at least one RACF group but can be part of several groups. Groups can be thought of as a hierarchical tree structure. Each group on the tree has a child or parent group or both. A parent group can be thought of as a superior group. A group hierarchy is only for group management. A superior group is the owner/administrator of the subgroup, but there is no inheritance of privileges. Within the scope of WebSphere, groups might control access to application servers, back-end data, methods within applications, and many other areas.

## **Resource classes**

When a program accesses an object that is protected at the operating system level, an authorization check is performed by the security manager, RACF in this case. Typical z/OS objects that would be accessed are MVS™ data sets, USS

file system objects, connections, and TCP/IP ports. In WebSphere Application Server, ACEEs are associated to the region's STCs and their address space. A profile can be defined for all resources and these profiles can control access to these resources.

Resources are controlled by a resource manager, which is responsible for calls (using SAF) for authorization checks. These resources also have related classes, which have to be activated. After a resource class has been activated, the resources manager will recognize that it is active through return codes from access control validation and start learning the correct security information.

## Auditing

With any security program, it is important to be able to perform an audit of accesses and violations of subjects. These audits are both for security reasons and change control processes. RACF has the ability to audit and report what resources are being accessed and by whom by creating SMF data records. Auditing tools can take this SMF data and create reports that can be analyzed and processed. In the scope of this book, auditing is important because sometimes accountability requirements make it necessary for user's credentials to be used from the front all the way through to the backend.

## System Authorization Facility (SAF)

RACF is fully compatible with SAF. All security checking requested of RACF is routed to RACF through SAF. The RACROUTE macro is used to invoke a SAF call to RACF for an authentication or authorization decision. Figure B-2 on page 95 describes the relationship between RACF and SAF.

# z/OS Security Server and SAF

RACF is invoked by resource managers at control points, typically using SAF interfaces.

## ■ Examples of resource managers:

- ▶ UNIX System Services
- ▶ Contents Supervisor
- ▶ DFP (OPEN,SCRATCH)
- ▶ UTILITIES
- ▶ Catalog Management
- ▶ VSAM
- ▶ AMS
- ▶ DFSMS
- ▶ IMS
- ▶ CICS
- ▶ TSO
- ▶ DB2
- ▶ JES2 and JES3
- ▶ Console Services
- ▶ PSF
- ▶ VTAM
- ▶ SDSF
- ▶ WebSphere
- ▶ And more...

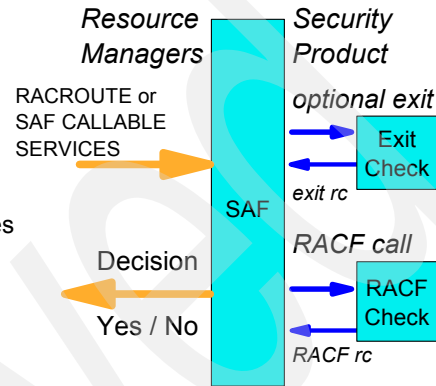


Figure B-2 Relationship between z/OS Security Server and SAF

## User IDs

Each system user is associated to a specific RACF user ID. Each user ID is validated against a secure password and can have attributes assigned that designate the user as a security administrator, data storage administrator, auditor, TSO user, CICS user, UNIX System Services (USS), or access to z/OS at certain days of the week or time of day.

## Certificate Name Filtering

RACF provides a process for administering large number of users without storing all of their digital certificates. RACF can selectively map users to either a unique user ID or a common user ID with auditability. RACF can associate many certificates with one user ID based on rules concerning portions of subjects or the issuer's distinguished name (DN) in the certificate.

## RACF classes

This section describes the pertinent RACF classes that are applicable to implementing WebSphere Application Server for z/OS V6 on z/OS. These classes represent a small subset of all available classes in RACF.

### APPL

Used for VTAM® application ID (APPLID) access. The APPL class controls access to applications. Kerberos is set up as an application, required so that the users can use the **kpasswd** command.

### CBIND

Controls the client's ability to bind to the server. WebSphere uses the CBIND class to control access to the server.

### DIGTCERT

Contains digital certificates and related information.

### DSNR

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. Figure B-3 on page 97 gives an overview of the EJBROLE and GEJBROLE classes in relation to J2EE roles.

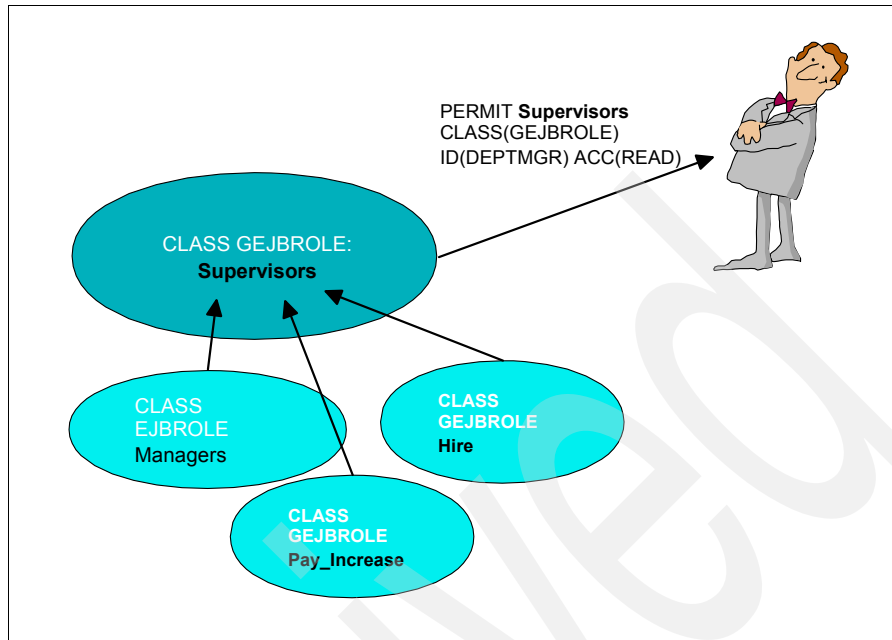


Figure B-3 RACF EJBROLE class and J2EE roles

## FACILITY

Miscellaneous uses; profiles are defined in this class so that resource managers can check users' access to the profiles when the users take some action. Here is where we place profiles for Digital Certificate, DCE, and Kerberos, plus UNIX System Services profiles (for example, BPX.DAEMON).

## KERBLINK

Mapping class for user identities of local and foreign principals. Used in Kerberos to map a unique RACF user ID to each foreign principal.

## LOGSTRM

Controls which applications can access the system logger resources.

## PTKTDATA

Passticket key class enables the Security Administrator to associate an RACF-secured signon secret key with a particular mainframe application that uses RACF for user authentication. Required for Kerberos setup, but no profile has to be created in this class, just activated.

## **REALM**

Used to define the local and foreign realms. Required for Kerberos setup, as you have to define your local realm to this class. You also use this class to define foreign realms, such as another z/OS.

## **SERVER**

Controls the server's ability to register with the daemon. The class is used in WebSphere to control whether a servant can call authorized programs in the controller.

## **SERVAUTH**

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 are using this class, you must give WebSphere and Kerberos access.

## **STARTED**

Used for identifying authorized system started procedures. WebSphere Application Server normally starts as a system task and would need an entry in the STARTED class to associate a valid RACF user ID and connected group to be able to access protected resources. Used in preference to the started procedures table to assign an identity during the processing of an MVS START command. For example, WebSphere, Kerberos are defined as started tasks in this profile.

## **SURROGAT**

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.

# **Back-end systems (EIS)**

Back-end systems (also referred to Enterprise Information Systems, or EIS) provide the data warehouse, storage, and primary data sources that are accessible by users of WebSphere Application Server.

## **DB2**

### **Primary authorization**

This is the user ID DB2 associates the incoming process.



## **Secondary authorization**

This is an additional user ID, used exclusively in DB2, that obtains access to DB2 objects (resources). This equates closely to an RACF group. Users who are connected to an RACF group can be correlated by DB2 as an equivalent secondary authorization ID and DB2 would allow access already defined for the secondary authorization ID by the DB2 database administrators.

## **DB2 objects**

These are resources that can be protected by DB2. These objects can be secured entirely within DB2 permissions, or via external security calls to RACF using the DB2 xDSNxxx classes.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 102. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *WebSphere for z/OS V6 Connectivity Handbook*, SG24-7064
- ▶ *z/OS WebSphere and J2EE Security Handbook*, SG24-6846

## Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS Cryptographic Service System Secure Sockets Layer Programming*, SC24-5901
- ▶ *z/OS HTTP Server Planning, Installing, and Using*, SC34-4826
- ▶ *z/OS Migration*, GA22-7499
- ▶ *z/OS Program Directory*, GI10-0670
- ▶ *z/OS Security Server LDAP Server Administration and Use*, SC24-5923
- ▶ *z/OS Security Server RACF Callable Services*, SA22-7691
- ▶ *z/OS Security Server RACF Command Language Reference*, SA22-7687
- ▶ *z/OS Security Server RACF Macros and Interfaces*, SA22-7682
- ▶ *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- ▶ *z/OS WebSphere and J2EE Security Handbook*, SG24-6846-01
- ▶ *z/OS Security Server RACF System Programmer's Guide*, SA22-7681

## Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ Java description of RMI-IIOP  
<http://java.sun.com/products/rmi-iiop/>
- ▶ Network Authentication Service (Kerberos)  
<http://web.mit.edu/kerberos/www/>
- ▶ Sarbanes-Oxley: Financial and Accounting Disclosure Information (HR 3763)  
<http://www.sarbanes-oxley.com/>
- ▶ Single sign-on (SSO)  
<http://www.opengroup.org/security/sso/>
- ▶ Benjamin, T. *Java Security on z/OS: An Introduction* (SHARE Session 1775), March 2002.  
[ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/pdf/share\\_03\\_2002\\_java\\_security\\_zos.pdf](ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/pdf/share_03_2002_java_security_zos.pdf)
- ▶ Kappeler, P. *z/OS LDAP and Security (as of z/OS 1.5)*. Guide/Share MVS, April 2004.  
<http://www.gsefr.org/compterendus/mvs/20040401/1avril-2ldap.pdf>

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## A

Accessibility 49  
APPL class 96  
Architectural suggestions 47  
Asserted identity 37  
Auditability 50  
Authentication 47  
Authorization 49

## B

Basic authentication 36  
BBOSBRAK 42  
binding 77  
BPX.DEFAULT.USER 48

## C

CBIND class 96  
Certificate-based authentication 36  
CICS 2  
Common Secure Interoperability (CSlv2) 19  
Corporate requirements 49–50  
credential 26  
credentials 51

## D

DB2 2  
declarative security 78  
Demilitarized Zone (DMZ) 34  
deployment descriptor 78  
DIGTCERT class 96  
Distributed Computing Environment (DCE) 90  
DSNR class 96

## E

EJBROLE class 43, 47, 96  
Enterprise Identity Mapping (EIM) 90

## F

FACILITY class 97  
FIRSTHIT 45

## G

GEJBROLE class 43, 47, 96  
Global security 43

## H

HTTP header 17

## I

IBM Directory Integrator (IDI) 46  
IMS 2  
Integrated Cryptographic Services Facility (ICSF) 89

## J

J2EE 45  
    security roles 77  
Java 2 security 72  
Java Authentication and Authorization (JAAS) 76, 78  
Java Authorization Contract for Containers (JACC) 17, 72, 82

## K

Kerberos 91  
Kerberos authentication 37  
KERBLINK class 97

## L

LDAP 2, 43, 46  
    distinguished name 44  
    native authentication 44  
    overview 91  
Lightweight Third Party Authentication (LTPA) 42  
LocalOS 43  
LOGSTRM class 97  
LTPA token 17

## M

MQSeries 2

## **N**

naming conventions 46  
native authentication 16, 46  
Network Address Translation (NAT) 34

## **O**

OMVS 48  
Open Cryptographic Enhanced Plug-Ins (OCEP) 91  
Open Cryptographic Services Framework (OCSF) 90

## **P**

passticket 88, 97  
passwords 48  
    requirements 48  
permission-based model 78  
practices 47  
principal 26  
Procedures 47  
programmatic security 78  
protocol 45  
PTKTDATA class 97  
Public Key Infrastructure (PKI) 90

## **R**

RACF  
    classes 96  
    groups 49  
    resources 50  
    *See also* z/OS Security Server  
RACO 48  
realm 27  
REALM class 98  
Reauthentication 48  
Redbooks Web site 102  
    Contact us xi  
Resource Access Control Facility (RACF) 89  
reverse proxy 35  
role 79  
role-based security design 49  
RunAs 79

## **S**

Sarbanes-Oxley 50  
scenarios 2  
SDBM 46

Security Attribute Service (zSAS) 19  
security domain 27, 44  
security policy domain 27  
SERVAUTH class 98  
SERVER class 98  
single sign-on (SSO) 26, 49  
SSL 45  
STARTED class 98  
subject 27  
SURROGAT class 98

## **T**

Tivoli Access Manager 46  
Trust Association Interceptor (TAI) 17

## **U**

UNIX System Services 47  
user IDs 47  
user registry 46  
UserRegistry interface 43

## **W**

WebSphere Application Server  
    administrative console 45  
WebSphere Global Security 45  
WebSphere V6 Security Requirements 1, 25  
WSCredential 76  
WSPrincipal 76  
WS-Security 72

## **Z**

z/OS HTTP Server 45  
z/OS Security Server (RACF)  
    components 88  
    industry standards 92





**Redpaper**

# WebSphere Application Server on z/OS and Security Integration

## **z/OS security summary**

This IBM Redpaper addresses the need for information in the area of integrating security between WebSphere Application Server on z/OS and the outside world.

## **Security integration scenarios**

In most cases, multiple security registries exist within a company with a different scheme of identities. This is even more likely in companies using z/OS. There are basically two “worlds”: the z/OS (RACF) world in which identities and their authorizations are kept in RACF and the outside world where identities and their authorizations are kept in LDAP, Microsoft Active Directory, or equivalent solutions.

## **Bridged security scenario**

In an e-business environment, the first authentication of a user is usually already performed before a request reaches the z/OS environment based on an ID not known in that exact form on z/OS. There are basically two challenges, and both of them are addressed in this paper:

- Authenticate a user on a distributed server and be able to trust that user when coming into WebSphere Application Server on z/OS.
- Propagate the user ID and eventual security credentials from the distributed environment to WebSphere Application Server on z/OS, and eventually transform the ID and credentials to something that is administered and understood on z/OS.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

## **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)