

Reduce Risk and Improve Security on IBM Mainframes: Volume 3 Mainframe Subsystem and Application Security

Axel Buecker

Marcela Kanke

Mohit Mohanan

Vinicius Oliveira

Vinodkumar Ramalingam

David Rowley

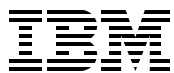
Botrous Thalouth

Jan Thielmann



Security

z Systems



International Technical Support Organization

**Reduce Risk and Improve Security on IBM Mainframes:
Vol. 3 Mainframe Subsystem and Application Security**

November 2015

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

First Edition (November 2015)

This edition applies to IBM z13 systems.

© Copyright International Business Machines Corporation 2015. 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

Notices	vii
Trademarks	viii
IBM Redbooks promotions	ix
Preface	xi
Authors	xi
Now you can become a published author, too	xii
Comments welcome	xiii
Stay connected to IBM Redbooks	xiii
Chapter 1. Introduction to major mainframe middleware components	1
1.1 Major software infrastructure on z/OS	2
1.1.1 Scope of this book	2
1.1.2 Overview of major z/OS application middleware	3
1.1.3 Major z/OS infrastructure middleware overview	5
1.1.4 Logical architecture for middleware on z/OS	7
1.1.5 Interfaces and intercommunication	13
1.2 Middleware security	14
1.2.1 Self-managed	14
1.2.2 External security manager	15
1.2.3 Exits	16
1.2.4 Audit and compliance reporting	17
1.3 Considerations for programming, configuration, and performance	18
1.4 Logging	19
1.4.1 Internal logging and the syslog	19
1.4.2 SMF	20
Chapter 2. Database managers	23
2.1 IBM DB2 for z/OS	24
2.1.1 Security concepts and architecture	24
2.1.2 Guidelines for configuring security	40
2.2 IBM Information Management System	41
2.2.1 Security concepts and architecture	41
2.2.2 Guidelines for configuring security	51
2.3 Virtual Storage Access Method	52
2.3.1 Security concepts and architecture	53
2.3.2 Guidelines for configuring security	57
Chapter 3. WebSphere Application Servers and web servers	59
3.1 IBM WebSphere Application Server overview	60
3.2 Security concepts and architecture	62
3.2.1 Global security configuration	63
3.2.2 SSL/TLS	75
3.2.3 Java security	77
3.3 Interfaces (transaction systems, databases, IBM MQ, web server, and other adapters)	80
3.3.1 WebSphere Message Queue	80
3.3.2 Event monitoring and recording (SMF, internal logging)	82

3.4 Guiding principles for configuring security	82
3.4.1 Common misconfigurations.	82
3.4.2 Security considerations.	83
Chapter 4. Transaction processing systems	91
4.1 IBM CICS Transaction Server.	91
4.1.1 Security concepts and architecture.	92
4.1.2 Guiding principles for configuring security	112
4.2 IBM Information Management System Transaction Manager.	114
4.2.1 Security concepts and architecture.	114
4.2.2 Guiding principles for configuring security	125
Chapter 5. IBM MQ messaging system	127
5.1 IBM MQ security concepts and architecture	128
5.1.1 Security setup	128
5.1.2 IBM MQ RACF RESLEVEL profile	133
5.1.3 IBM MQ resource security.	135
5.1.4 IBM MQ Security Management.	139
5.1.5 IBM MQ CICS adapter	140
5.1.6 IBM MQ IMS adapter	141
5.1.7 Channel security	141
5.1.8 Threats and risks	142
5.1.9 Event monitoring and recording	142
5.2 Guiding principles for configuring security	144
5.2.1 Common misconfigurations.	144
5.2.2 Security considerations.	145
Chapter 6. Session management.	147
6.1 IBM Session Manager basics	148
6.2 Security concepts and architecture	148
6.2.1 User authentication	150
6.2.2 Static menus	156
6.2.3 Security setup	158
6.2.4 Session Manager commands	162
6.2.5 Session Manager command statements.	163
6.2.6 Threats and risks	163
6.2.7 Event monitoring and recording	164
6.3 Guiding principles for configuring security	164
6.3.1 Certificate express logon and PassTickets.	164
6.3.2 Security considerations.	165
6.3.3 VTAM dump considerations	165
6.3.4 Protection of IBM Session Manager libraries and PDSE data sets.	165
Chapter 7. Scheduling systems.	167
7.1 Tivoli Workload Scheduler for z/OS basics	168
7.2 Security concepts and architecture	168
7.2.1 Protecting the Workload Scheduler subsystem	169
7.2.2 Controlling access to Workload Scheduler	169
7.2.3 Event-triggered tracking	172
7.2.4 Security exits.	172
7.2.5 Threats and risks	173
7.2.6 Event monitoring and recording	173
7.3 Guiding principles for configuring security	173
7.3.1 Workload Scheduler applications	173

7.3.2 Submitting user IDs.	174
7.3.3 Protecting JES resources	175
7.3.4 Batch and protection of source JCL and code	176
Related publications	179
IBM Redbooks	179
Help from IBM	179

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 grant 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 illustrate 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.

Trademarks

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

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

AIX®	IMS™	S/390®
CICS®	InfoSphere®	System Storage®
DataPower®	Language Environment®	Tivoli®
DB2®	MVS™	VTAM®
DRDA®	NetView®	WebSphere®
DS8000®	OMEGAMON®	z Systems™
Global Business Services®	Parallel Sysplex®	z/OS®
Global Technology Services®	RACF®	z/VM®
Guardium®	Rational®	z/VSE®
IBM®	Redbooks®	zEnterprise®
IBM z™	Redbooks (logo)  ®	zSecure™
IBM z Systems™	S-TAP®	

The following terms are trademarks of other companies:

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

Linear Tape-Open, LTO, Ultrium, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

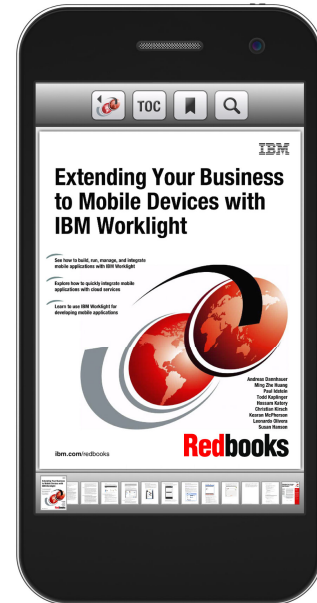
Get the latest version of the Redbooks Mobile App



Download
Now

SOI

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication documents the strength and value of the IBM security strategy with IBM z™ Systems hardware and software. In an age of increasing security consciousness and more and more dangerous advanced persistent threats, IBM z Systems™ provides the capabilities to address the needs of today's business security challenges. This publication explores how z Systems hardware is designed to provide integrity, process isolation, and cryptographic capability to help address security requirements. We highlight the features of IBM z/OS® and other operating systems, which offer a variety of customizable security elements. We discuss z/OS and other operating systems and additional software that use the building blocks of z Systems hardware to provide solutions to business security needs. We also explore the perspective from the view of an enterprise security architect and how a modern mainframe has to fit into an overarching enterprise security architecture.

This book is part of a three-volume series that focuses on guiding principles for optimized mainframe security configuration within a holistic enterprise security architecture. The series' intended audience includes enterprise security architects, planners, and managers who are interested in exploring how the security design and features of z Systems, the z/OS operating system, and associated software address current issues such as data encryption, authentication, authorization, network security, auditing, ease of security administration, and monitoring.

Authors

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

Axel Buecker is a Certified Consulting Software IT Specialist at the ITSO Austin Center. He writes extensively and teaches IBM classes worldwide about areas of software security architecture and network computing technologies. He holds a degree in Computer Science from the University of Bremen, Germany. He has 29 years of experience in various areas that are related to workstation and systems management, network computing, and business solutions. Before he joined the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

Marcela Kanke is a senior consultant for Produban, Santander, in the São Paulo, Brazil area.

Marcela Kanke is an Application Architect with the IBM Global Business Services® division. Mohit has 14 years of experience in mainframe application design, development, and validation. He holds a bachelor's degree in Computer Science from the University of Kerala, India. His current areas of interest are mainframe transaction processing architectures, security architectures, and integration architectures.

Mohit Mohanan is a Level 1 IT Specialist in Belo Horizonte, Brazil. He has 10 years of experience in IT solutions. He joined IBM in 2009 as an IT Specialist for IGA Canada, supporting internal IBM accounts as a member of the technical leadership team, with a focus on infrastructure. He specializes in infrastructure, middleware, and server support. Vinicius is now a team leader for the IBM Rational® Support Team in Brazil. His previous work experience includes network administration and analysis.

Vinicius Oliveira is a Certified Senior IT Specialist for IBM z Systems. He works on architecture and technical solution development for the Mainframe Speciality Services Area in IBM Global Technology Services® Delivery Technology and Engineering team. He has an M.Phil degree in Computer Science, a master's in information technology, and an MBA. He also has a post-graduate diploma in Cyber Law. He has more than 14 years of experience on z Systems and has had various roles in IBM since 2004.

Vinodkumar Ramalingam has been working at IBM, predominantly on the Security team, for more than 16 years in the small community of Ballarat, Australia. Before IBM, he spent several years as a programmer, working for both a large company and privately. He has written code in several languages for more than 30 years.

David Rowley is an IBM S/390® Development Consultant with the Cairo Technology Development Center. He provides worldwide second- and third-level support for SM/370, text search engine and text extenders, z/OS, and DB/DC. He has more than 35 years of experience with IBM mainframes, starting with IBM System/360 as an application and system programmer. He also worked in the IBM Scientific Center. While he was there, he designed and implemented an Arabic morphology system, the algorithm of which is used by many IT companies. Botrous provides support for IBM clients in the Arabian Gulf area and teaches many courses about z/OS, IBM z/VM®, IBM z/VSE®, Linux OS on IBM z Systems, IBM CICS®, IBM DB2®, and application programming. He has a Bachelor of Engineering degree from Cairo University.

Botrous Thalouth is a Certified IT Specialist for IBM z/OS in IBM Software Group, Germany. He is member of a cross-brand z Systems Software Services Team, One of his main focus areas is security on z/OS, mainly IBM RACF® with IBM Security zSecure™, but he also works on other products in that area. He delivers service projects for clients across Europe and presents regularly on IBM z Systems conferences. He is 11 years younger than RACF and holds a Bachelor of Science degree in Applied Computer Science. Jan joined IBM in 2006 and started working with z Systems and z/OS in 2008. He has been as a member of the IBM z Systems Software Services Team since 2009.

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

Lydia Parziale

IBM International Technical Support Organization

Kerstin Ackermann, Julie Bergh, Leigh Compton, Lennie Dymoke-Bradshaw, Morag Hughson, Mark Nelson, Jamie Pease, Richard Schneider, Peter Siddell, Maida Snapper, Doug Specht, Joerg-Ulrich Vesper, Nigel Williams, Holger Wunderlich
IBM

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

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

- Use the online **Contact us** review Redbooks form:

ibm.com/redbooks

- Send your comments by email:

redbooks@us.ibm.com

- Mail your comments:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Introduction to major mainframe middleware components

In this introduction, we cover security concepts and provide an architectural overview of major software components of IBM z/OS operating system.

This chapter covers the following topics:

- ▶ Major software infrastructure on z/OS
- ▶ Middleware security
- ▶ Considerations for programming, configuration, and performance
- ▶ Logging

1.1 Major software infrastructure on z/OS

This chapter introduces the major z/OS software components. We discuss software packages that are deployed in most z/OS installations. The set of products that we describe in this book is not complete, so you might miss some of the products that are used in your installation. But the team that wrote this book needed to decide on a subset of available software on z/OS to include in the book. In this chapter, we cover basic topics on software use in z/OS. We describe common concepts and architecture that is valid for all products that are covered in this book. The subsequent chapters include detailed descriptions and guiding principles for all the products mentioned in this introduction.

1.1.1 Scope of this book

In this section, we define the terms that we used as guidelines to include software packages in this book. The most common term to describe major software components in z/OS is to call them *subsystems*. However, we do not want to use that term in this book, because the meaning of the word *subsystem* varies with the context of the subject.

A common definition of what a subsystem is to z/OS, on a technical level, can be found in *MVS Using the Subsystem Interface*, SA38-0679-00:

“A subsystem is a service provider that performs one function or many functions but does nothing until it is requested... a subsystem must be the master subsystem or be defined to IBM MVS™ in one of the following ways...”

As you can see, the technical definition of what a subsystem is to z/OS is not satisfactory for the purposes of this publication. The term opens a wide scope of what to include in the discussion.

We prefer to use the term *middleware* to refer to major software components in z/OS. Typically, middleware is software that offers services to applications that the operating system does not provide. A database manager, for example, is middleware that offers services for organized collection and storage of data. This function is not provided by the operating system or, to be more specific, is not provided to the extent that is needed for data management in organizations.

In this book, we include major middle products that are not elements of z/OS but are necessary to operate a z/OS production environment. The two other volumes in this three-book series include related information:

- ▶ The products that are elements of z/OS (such as JES2) are discussed in *Reduce Risk and Improve Security on IBM Mainframes: Volume 1 Architecture and Platform Security*, SG24-7803.
- ▶ The z/OS Communication Server element of z/OS is discussed in *Reduce Risk and Improve Security on IBM Mainframes: Volume 2 Mainframe Communication and Networking Security*, SG24-8195.

For this introduction, we need to differentiate two more types of products that are covered in this book:

- ▶ We call the software used to support the applications running on z/OS *application middleware*. This includes all products that are used to run business applications and to manage the data that is used in these applications. Databases, transaction systems, web servers, application servers, and messaging systems are included in this definition.

- The second family of middleware products is used to manage the computing infrastructure in a z/OS environment. This includes session management systems for accessing the mainframe, scheduling systems for work distribution throughout the infrastructure, language environment, systems management, automation, monitoring products, and storage management software. These products are summarized under the term *infrastructure middleware*.

1.1.2 Overview of major z/OS application middleware

This section provides a short introduction to the selection of major application middleware products covered in this book.

IBM DB2

IBM DB2 is a relational database system with a proven record in availability, scalability, and reliability. It is now optimized for service-oriented architecture (SOA), customer relationship management (CRM) systems, and data warehousing.

IBM DB2 for z/OS is the enterprise data server, designed and tightly integrated with the IBM z Systems mainframe to use the strengths of the system and reduce total cost of ownership through process enhancements and productivity improvements for database administrators and application developers.

New structures, such as the ability to change data definitions without disrupting online performance, continue to enhance availability and scalability in DB2, and bottlenecks have been removed to ensure optimal performance. DB2 for z/OS provides the core infrastructure for web business applications and growing data management needs (big data).

The security capabilities of DB2, together with guiding principles for security, are described in Chapter 2, “Database managers” on page 23.

IBM IMS

IBM Information Management System (IMS™) is a transaction and hierarchical database management system with a long-standing reputation for superior availability, performance, capacity, and integrity for critical online data and applications. It includes two major components:

- The IMS database manager

The IMS Database Manager offers a central point for control and access to application data. It includes a full set of utility programs to provide all of these functions within the IMS product.

- The IMS transaction manager

The IMS Transaction Manager gives users access to applications running under IMS. The users can be people at terminals or workstations, or they can be other applications on the same z/OS system, on other z/OS systems, or on non-z/OS platforms.

The security capabilities of IMS and the guiding principles for security, are described in Chapter 2, “Database managers” on page 23, and Chapter 4, “Transaction processing systems” on page 91.

IBM CICS Transaction Server

IBM supplies the total infrastructure that companies need to model, develop, deploy, and manage their critical business applications. As part of this infrastructure, IBM CICS TS uses the qualities of service associated with the z/OS platform.

CICS provides a complete solution to enable management of resources through a single point of control that is consistent throughout the enterprise. CICS enables execution of complex and demanding mixed-language application workloads while efficiently integrating with enterprise service-oriented architecture (SOA). It provides a runtime container for applications in COBOL, PL/I, Assembler, C, C++, and Java, so it is a language-neutral approach. This ensures that existing skills can be used to build and deploy new applications. CICS provides open-standards-based connectivity that is consistent with the preferred web services implementation while preserving the expected qualities of service of the trusted applications, all at an unparalleled cost per transaction.

The security capabilities of CICS are described in detail in Chapter 4, “Transaction processing systems” on page 91.

IBM WebSphere Application Server for z/OS

IBM WebSphere® Application Server is a comprehensive, sophisticated, Java 2 Enterprise Edition (J2EE) and web services technology-based application system. WebSphere Application Server on z/OS is a Java Platform, Enterprise Edition implementation that conforms to the current Software Development Kit (SDK) specification, supporting applications at an API level. As mentioned, it is a Java Application deployment and runtime environment built on open standards-based technology that supports all major functions, such as servlets, JavaServer Pages (JSPs), and Enterprise JavaBeans (EJBs), including the latest technology integration of services and interfaces.

The application server run time is highly integrated with all inherent features and services offered on z/OS. The application server can interact with all major subsystems on the operating system, including DB2, CICS, and IMS. It has extensive features for security, performance, scalability, and recovery. The application server also uses sophisticated administration and functions to provide seamless integration into any data center or server environment. It is built on open standards-based technology and supports all Java APIs needed for Java Platform compliance.

The security capabilities of WebSphere Application Server for z/OS are described in Chapter 3, “WebSphere Application Servers and web servers” on page 59.

IBM HTTP Server on z/OS

Users of z/OS for the past several years have had a choice of two HTTP servers. The original HTTP server for use on z/OS, introduced in the 1990s, was often referred to as the “Domino Go Web server” (DGW) or simply “IHS” (for IBM HTTP Server). It serves static and dynamic web pages and has the same capabilities as any other web server, but it also has features that are specific to the z/OS operating system.

When WebSphere Application Server became available on z/OS, around 2003, this included an HTTP server based on the widely used Apache HTTP server. IBM HTTP Server now installs with the base operating system on z/OS V2R2 and later. No separate installation is required.

IBM HTTP Server is based on Apache HTTP Server 2.2.8, with additional fixes. The original Apache server was developed in 1995 and is now developed and maintained by the Apache Software Foundation, an open source community. Apache is a C language implementation of an HTTP web server. It is widely used on many operating systems and has a broad user community. You can extend the core capability by developing your own modules or using modules already developed.

IBM MQ

IBM MQ (formerly known as WebSphere MQ) is a family of products that provide message queuing capabilities for z Systems applications and for applications on many other platforms. It facilitates the secure and reliable exchange of information between applications, systems, services and file by sending and receiving message data via messaging queues. Applications may reside on the same platform or on different computing environments that are physically separated by large distances. It is supported on the following platforms:

- ▶ IBM z/OS
- ▶ Linux OS on IBM z Systems and other platforms
- ▶ Variations of UNIX (such as HP-UX, IBM AIX®, and Solaris)
- ▶ Microsoft Windows

MQ provides queue managers for connection. Communication paths are called *channels*. The channels can use a variety of protocols, including SNA. However, in more recent times, most channels use TCP/IP.

The security capabilities of IBM MQ are described in Chapter 5, “IBM MQ messaging system” on page 127.

1.1.3 Major z/OS infrastructure middleware overview

The most important mission of a job scheduler is to provide a fully automated system that efficiently uses system resources to process the production workload in the least amount of time, with virtually no wasted resources. IBM Tivoli® Workload Scheduler does this by automating, monitoring, and controlling the flow of production work through the enterprise’s entire IT infrastructure.

IBM Tivoli Workload Scheduler for z/OS

IBM Tivoli Workload Scheduler automates, monitors, and controls the flow of production work through an enterprise’s entire IT infrastructure. From a single point of control, it analyzes the status of the production work and schedules the processing of the workload according to business policies. It supports a multiple user environment and enables distributed processing. Tivoli Workload Scheduler is the job scheduler product from IBM. It consists of two separate products:

- ▶ IBM Tivoli Workload Scheduler for scheduling workloads on distributed platforms, such as Windows, UNIX, Linux, and other open system environments
- ▶ IBM Tivoli Workload Scheduler for z/OS for managing workloads on IBM z Systems mainframes

Although the products are separate, they can be integrated to provide a true end-to-end job scheduler. This solution can manage a production workload from a centralized single point of control and allow workloads to be controlled on virtually every platform found in today’s IT environments.

The security capabilities of Tivoli Workload Scheduler for z/OS are described in Chapter 7, “Scheduling systems” on page 167.

Session Manager

IBM Session Manager provides secure and reliable access to multiple applications from a single 3270 terminal session. The session can exist on either a Systems Network Architecture (SNA) or a TCP/IP network, and that terminal session can simultaneously connect to and navigate between multiple Virtual Telecommunication Access Method (IBM VTAM®) and TCP/IP applications.

The security capabilities of IBM Session Manager are described in Chapter 6, “Session management” on page 147.

DFSMS hierarchical storage manager

DFSMSHsm is a disk storage management and productivity product for managing low-activity and inactive data. It provides backup, recovery, migration, space management functions, and full-function disaster recovery support. DFSMSHsm improves disk use by automatically managing both space and data availability in a storage hierarchy.

- ▶ *Availability management* is used to make data available by automatically copying new and changed data set to backup volumes.
- ▶ *Space management* is used to manage DASD space by enabling inactive data sets to be moved off fast-access storage devices, so it creates free space or new allocations.

DFSMSHsm also provides other supporting functions that are essential to your installation's environment.

DFSMS removable media manager

In your enterprise, you store and manage your removable media in several types of media libraries. For example, in addition to your traditional tape library (a room with tapes, shelves, and drives), you might have several automated and manual tape libraries. You probably also have both onsite libraries and offsite storage locations, also known as vaults or stores.

With the DFSMS removable media manager (DFSMSrm) functional component of DFSMS, you can manage your removable media as one enterprise-wide library (single image) across systems. Because of the need for global control information, these systems must have access to shared DASD volumes. DFSMSrmm manages your installation's tape volumes and the data sets on those volumes. It also manages the shelves where volumes reside in all locations except in automated tape library data servers.

DFSMSrmm manages all tape media (such as cartridge system tapes and 3420 reels) and other removable media that you specify for it to manage. For example, DFSMSrmm can record the shelf location of optical disks and track their vital record status (but it does not manage the objects on optical disks).

IBM Security Key Lifecycle Manager for z/OS

IBM Security Key Lifecycle Manager for z/OS manages encryption keys for storage, simplifying deployment and maintaining availability to data at rest natively on the z Systems mainframe environment. It simplifies key management and reporting for compliance with privacy and security regulations and helps manage the growing volume of encryption keys across an organization with simplified deployment, configuration, and administration of key generation and key lifecycle management.

Security Key Lifecycle Manager centralizes key management for devices across an organization. It supports the encryption of IBM 3592 tape cartridges, IBM LTO tapes, and IBM DS800 disks. It can also simplify event logging through the use of the z/OS System Management Facility.

IBM Security Key Lifecycle Manager

In your enterprise, many symmetric keys, asymmetric keys, and certificates can exist. All of these keys and certificates must be managed. Key management can be handled either internally by an application, such as IBM Tivoli Storage Manager, or externally by an Encryption Key Manager (EKM). In this section, we describe IBM Security Key Lifecycle Manager.

Security Key Lifecycle Manager performs key management tasks for IBM hardware that is encryption-enabled, such as TS1120 and TS1130 Tape Drives and Linear Tape-Open (LTO) Ultrium 4 Tape Drives. The tasks provide, protect, store, and maintain encryption keys that encrypt information being written to and decrypt information being read from tape media. Security Key Lifecycle Manager operates on a variety of systems.

It is a shared resource that is deployed in several locations within an enterprise. It is capable of serving numerous IBM encryption tape drives, regardless of where those drives reside (for example, in tape library subsystems, connected to mainframe systems through various types of channel connections, or installed in other computing systems).

In contrast, the sole task of the Tivoli Lifecycle Key Manager is to handle the serving of keys to the encrypting tape drives. The Security Key Lifecycle Manager does not perform any cryptographic operations, such as generating encryption keys, and it does not provide storage for keys and certificates. To perform these tasks, Security Key Lifecycle Manager rely on external components.

1.1.4 Logical architecture for middleware on z/OS

This section provides an overview of middleware implementation on z/OS and common approaches.

Overview of application middleware

First, let's review application architecture concepts in z/OS concepts, because those illustrate how work is entered into the system and is then processed. This is not an in-depth discussion of how to set up application architecture on z/OS, but it makes you aware of the broad spectrum of data entry and data processing concepts. That makes it easier to understand why a wide range of aspects must be taken into account when designing security concepts for your applications.

Traditionally, work was entered into a mainframe system through the 3270 terminal. This is still one of the most common and widely used ways of communicating with the mainframe, especially for administration and system management purposes. However, for enterprise applications and data processing, this entry point has lost importance steadily over the recent decades. Web technology has changed the way we interact with the mainframe, so we can now enter data in various ways. Application middleware products offer various concepts of connecting over the network by using TCP/IP and other protocols.

Today, a mainframe may be used as a central data hub within an enterprise architecture, but the business applications are not deployed on the mainframe. Instead, they reside in the distributed environment and use the mainframe's data management capabilities with DB2 or IMS DB, and others.

If you do use transaction systems on the mainframe, like CICS and IMS TS to host your business applications and processes, the user interaction is probably not implemented using 3270 terminals, but in application servers somewhere in your environment. These application servers communicate, using web or Java technology, with the transaction systems on the mainframe. Of course, the WebSphere Application Server on z/OS is widely used as a user interaction system in combination with back-end transaction systems and databases on the mainframe.

In z/OS, there are also various ways in which middleware and business applications communicate with each other. Messaging systems such as IBM MQ and cross-memory connections or cross-sysplex communication between address spaces are examples of inter-application communication in z/OS.

Traditional batch workloads play an important role for many customers. Although this particular approach is not quite the focus anymore, in contrast to more modern concepts such as web technology or big data, it is still crucial to many installations. Batch processing provides an efficient way to process huge amounts of business data. Batch processes can also use application middleware components, such as DB2 or IMS. Many batch applications rely on interfaces with the middleware, most of which are provided through utilities of the middleware products.

A typical batch application might use DB2 data to process contracts that have been closed by an insurance company on the same day, using IBM CICS, and create contract confirmation letters that are then printed and sent to the customers. Keeping such connections in mind, you can also find that there is an inherent relationship between traditional batch workloads and online transaction processing with its middleware products, possibly using web technology.

Business-critical data is used in various areas on the system, so its protection is crucial. The data needs to be properly protected on its way through the system by all participating parties. Considering the importance of the data on the mainframe, you need to take special care of it not only within boundaries of the middleware but also on the communication points between them. All of the middleware products implement security in a certain way. None of the concepts are identical, but all share common concepts.

We describe several security concepts in 1.2, "Middleware security" on page 14. The variety of concepts for implementing application middleware provides a considerable amount of flexibility and the chance to adapt to changes in your enterprise architecture.

Figure 1-1 illustrates the concepts that we described previously. A business application is deployed within IMS. Then, IMS communicates with DB2 for data exchange by using IBM MQ. RACF is always part of the processing for authorization purposes, with the need to maintain a user's identity throughout the operation, from the distributed environment to the mainframe and on the mainframe. A user has several possible ways to access the application via 3270 terminals, Java technology (JDBC) and application or web servers.

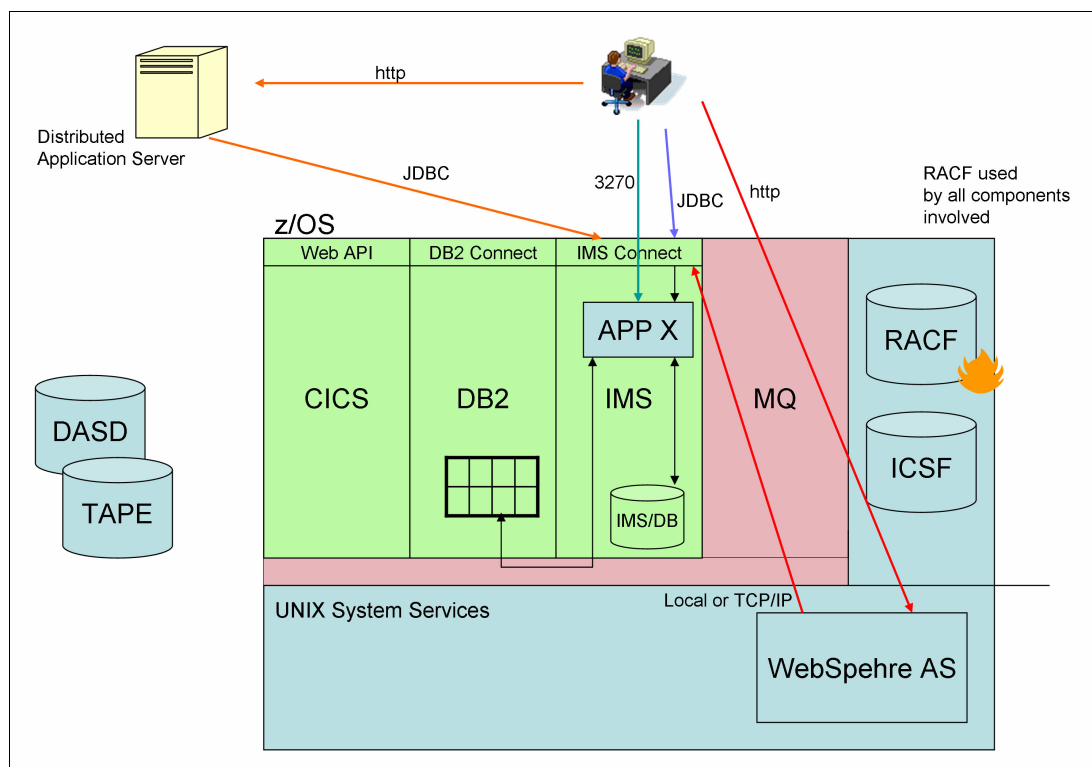


Figure 1-1 Application middleware concepts basic flow

Overview of infrastructure middleware

In the previous section, we gave an overview of the implementation of application middleware products in mainframe environments. We now provide an overview of the infrastructure middleware that is necessary to operate a mainframe environment and the applications in it. Again, this is just a summarization of various different possible implementation scenarios. For a detailed description of these topics, see 1.1.2, “Overview of major z/OS application middleware” on page 3 and 1.1.3, “Major z/OS infrastructure middleware overview” on page 5.

To achieve a usable, stable, and performant operation of z/OS systems, most customers use additional infrastructure middleware. These middleware products do not host applications that are running on the system, but they provide essential services to other products that do, to the operating system, or to the user. Infrastructure middleware also ensures that the system provides *operational security* by helping to ensure that a reliable and available environment is provided to critical applications on the mainframe.

Security as a holistic approach does not consist only of identity management and data security within an application. For example, it is important for operations staff to know how healthy the system is at any particular time. Performance shortages or bottlenecks in the I/O infrastructure might affect the applications running on the system. Therefore, monitoring and systems management software is used. IBM, among other vendors, offers a wide range of monitoring, systems management, and performance analysis tools for the mainframe.

In addition to proper monitoring, z/OS systems also need automation tools to minimize outages and ensure that the system operates in a productive state, with as little downtime as possible. Automation tools ensure that the system can be shut down and restarted quickly and that the correct processing sequence and intelligently respond to log messages and unfamiliar circumstances.

Without automation software, this would not be possible in modern mainframe environments. The effort it takes to shut down and IPL a sysplex manually, for example, is immense, if even possible at all. A message on the system log is rolled out on any screen within a fraction of a second, so there is literally no time to react manually. Automation tools also ensure the proper execution of software during run time and can react to messages that indicate software failures and provide mechanisms for software recovery. Some housekeeping functions are also performed by using automated tools.

Storage management is another important infrastructure middleware component in z/OS. The operating system provides access methods to the storage devices on disk and tape and defines the format and structure of data that is stored on them. Storage management software provides services to efficiently manage that data. On z/OS, this is done by using the IBM Data Facility Storage Management Services (DFSMS) family of products. DFSMS provides services to the user and the operating system for efficient management of data in z/OS. This includes the policy-driven decisions about where to store a data set (tape or disk and which devices in particular), backup and recovery solutions, data set management services, and tape management services.

We must also include the z/OS IBM Language Environment® in this discussion. Language Environment provides compilers for COBOL, Assembler, PL/1 languages on z/OS. Together with the Java infrastructure in UNIX System Services, it is an essential component for software development on the mainframe.

Users who are not accessing the system through web interfaces from remote systems (workstations or servers) must use traditional 3270 access via the Telnet 3270 server of the z/OS Communication Server component. This is still one of the most common ways to access the mainframe. To enable the users to be more productive, many customers implement session management software, such as IBM Session Manager for z/OS, between the user and the system. The users log on to the session manager first. The session manager then offers the user a selection of different sessions, based on the user's authorizations. The user can then choose one or more sessions, and the session manager handles the login.

Figure 1-2 summarizes and illustrates, at a high level, possible scenarios of infrastructure middleware use involved in hosting an application in modern mainframe environments. The application might be application middleware software, a program within the application middleware, or simply a user program in z/OS.

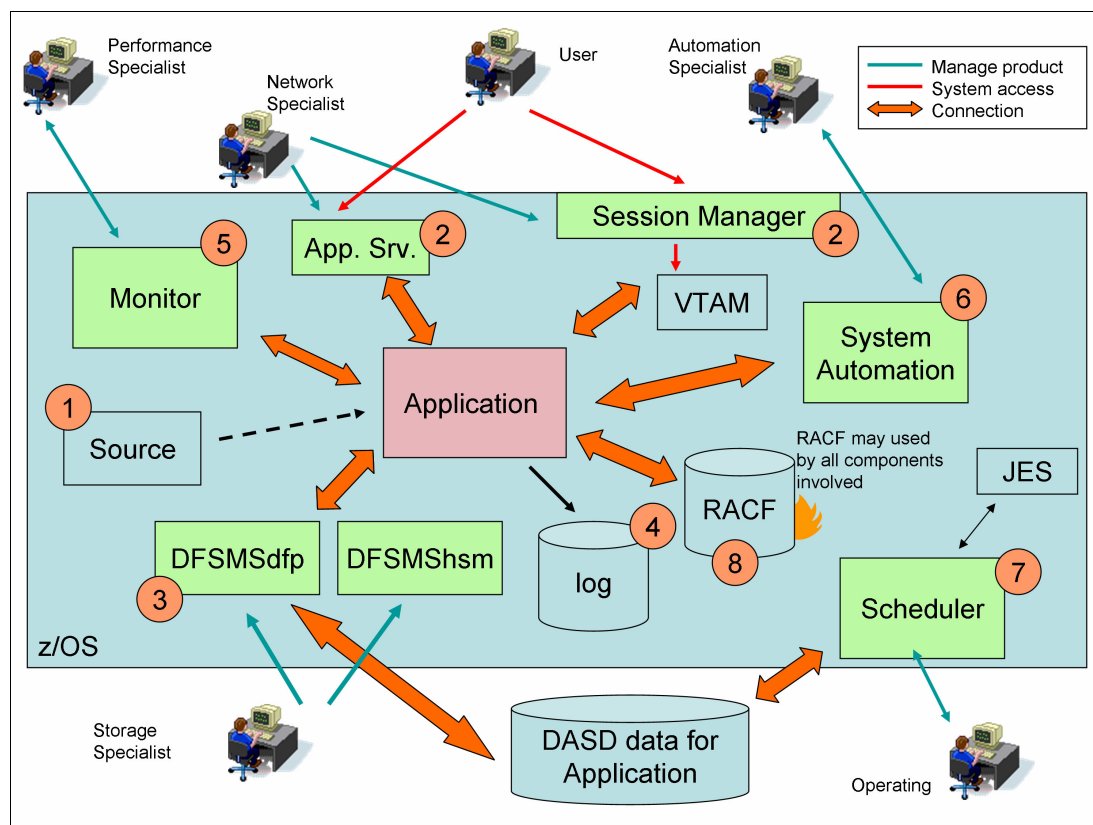


Figure 1-2 Infrastructure middleware concepts, basic flow

To summarize, consider what we described before but focus on just the infrastructure middleware involved:

1. Your application is either compiled from code or installed by using an installation source. When the executable code is in place, you start to use the application.
2. A user may be access the application directly, through the session manager on the system, or through an application server, which are maintained by a network specialist.
3. The application may store and read data on disk or tape. To allocate files and access the data, Data Facility Product (DFSMSdftp) is used. DFSMSShsm is used to manage data in terms of backup and space use. Storage management is handled by a storage specialist.
4. Your application may produce log data. For logging, it can use either internal mechanisms or SMF and the syslog.
5. The application can be important, so be sure to monitor it. Use monitoring solutions that use log and real-time data to display the health status of the application to a monitoring specialist.
6. Because your application is critical, you need to ensure its constant availability. Include it in your system automation process, which is maintained by an automation specialist. System automation also may directly influence the application through an operator command interface.

7. The data that your application produces while it is used during business hours needs to be processed at night or during non-prime business hours. Therefore, a scheduling system and the Job Entry Subsystem (JES) are also included in the diagram in Figure 1-2 on page 11. The scheduler manages the processing of application data based on rules defined by the operating staff.
8. Your application and infrastructure middleware products use RACF for authorization checks of users.

As you can see from the previous diagram, this all leads to a complex environment in which security implementation is not a trivial task. There are many roles and users involved, and even more interfaces and connections exist between the application and the middleware and between the middleware and the system. Data from the application flows through the system at various points and is processed not only by the application but also by infrastructure middleware.

Identity determination concerns modern IT environments

In today's heterogeneous computing environment, user interfaces and business processes are implemented outside of the mainframe environment, but work eventually runs within a z/OS subsystem, such as CICS or DB2.

Because the use of distributed identities that initiate work on the z/OS system is increasing, special care must be taken to correctly identify the identify of the user who initiated work on z/OS. The implementation of a complete audit trail is a fundamental problem when z/OS systems act as a backbone for distributed applications, processes, and servers. Often, technical user IDs are used for all identities that submit work from outside z/OS, which makes it almost impossible to audit the actual user responsible.

In a typical cases, a user initiates a transaction by using the distributed identity. When the transaction enters z/OS, the user's distributed identity is lost, because in the z/OS environment, only a z/OS security server RACF user ID is relevant. In common cases, transactions running in a z/OS subsystem (for example, CICS) execute under a single technical z/OS RACF user ID for work from outside the system, even though the users who are initiating the transactions are different. There is no association between the distributed identity of the user who starts the transaction and the RACF user ID under which the transaction runs.

A common approach to circumvent problems like these is to use z/OS Identity Propagation technology, introduced with z/OS version 1.11. For more information, see the IBM Redbooks publication titled *z/OS Identity Propagation*, SG24-7850.

1.1.5 Interfaces and intercommunication

This section describes the types of interfaces that middleware components use to establish communication with each other and the operating system.

First of all, middleware components on z/OS provide several interfaces to the outside world. Traditional terminal connections to the mainframe are still important to most customers, especially for administration purposes.

Application middleware products provide services to distributed systems on different levels. By using interfaces on a code level (such as JDBC and ODBC), distributed applications can communicate directly with the mainframe software. This provides a tight interaction between software on different systems. Many z/OS software products use such interfaces to communicate with other software on the mainframe. Even network interfaces are used for communication within the same z/OS instance. Many products also offer proprietary APIs to provide a communication interface with other software products on z/OS.

Users can interact directly with the system by using terminals or by using web technology that is provided by application and infrastructure middleware. Access is also provided using gateways. Gateway software resides either on the mainframe or on a distributed system and is accessed by users and applications. The gateway consolidates the access requests by the users, possibly provides security, and executes the requests on the mainframe software. The methods that it uses to access the mainframe may differ from the access methods used by the users and applications in front of the gateway.

All of these interfaces need security applied to them. The way for an attacker's entry into the mainframe leads through the interfaces that it provides to the outside world. Unsecured web services are easy for hackers to exploit, and the structure and concepts require no specific mainframe knowledge but provide ways to access mainframe data.

On the mainframe, software is not deployed in the way that it is on distributed systems. A middleware product that runs in z/OS is usually executed in an address space. The address space is a set of contiguous virtual addresses that are available to a program's instructions and its data. The range of virtual addresses that is available to a program starts at 0 and can go to the highest address permitted by the operating system architecture.

The use of address spaces allows z/OS to maintain the distinction between the programs and data that belongs to each address space. The private areas in one user's address space are isolated from the private areas in other address spaces, and this provides much of the operating system's security. Each address space also contains a common area that is accessible to every other address space. Because it maps all of the available addresses, an address space includes system code and data in addition to user code and data.

When the application starts, the address space is created and the application is loaded into it. During run time, the application and the data that it is processing are in the address space. Application middleware products, such as CICS, IMS, and DB2, each use several address spaces during run time, each assigned a different task.

Address spaces can communicate with each other. z/OS provides two methods of inter-address space communication:

- ▶ Scheduling a service request block (SRB), which is an asynchronous process
- ▶ Using cross-memory services and access registers, which is a synchronous process

A program uses an SRB to initiate a process in another address space or in the same address space. The SRB is asynchronous in nature and runs independently from the program that issues it, thereby improving the availability of resources in a multiprocessing environment. A program uses cross-memory services to access another user's address spaces directly.

z/OS cross-memory (XM) services require the issuing program to have special authority, which is controlled by the authorized program facility (APF). This method allows efficient and secure access to data owned by others, to data owned by the user but stored in another address space for convenience, and for rapid and secure communication with services, such as transaction managers and database managers. Cross-memory services are implemented by many z/OS subsystems and products. They can also be synchronous, enabling one program to provide services that are coordinated with other programs.

Application middleware is typically not deployed only once within a z/OS system. An organization might, for example, deploy several DB2 systems, each serving another major business application. The DB2 address spaces of one system communicate with each other and with the other DB2 cross-memory intercommunication methods or SRBs. The configuration of the separate instances may be generic for all instances up to a certain point and contain instance-specific parameters. The same holds for the security configuration of the instances.

1.2 Middleware security

In this section, we take a closer look at what security concepts are used by middleware components. We provide an overview of what security implementations are available on z/OS.

1.2.1 Self-managed

z/OS software may implement its own security. Self-managed security is one way to authorize users on a software functions and its data. When the security is self-managed, it is independent from the Tivoli Event Services Manager and may use its own security repository for user authorization purposes. The way in which a software implements security is defined by the software itself.

One problem with self-managed security is the probable interdependence between the security implemented by the product and the external security manager, such as Tivoli Event Services Manager. If your product stores authorization information within a data set, that data set needs to be protected by the Tivoli Event Services Manager. The same holds for any exit that you might use for self-managed security. The security of the exit is managed by the Event Service Manager, which protects the data sets where the source and the executable file of the exit resides. You are also likely to use the user ID from the Tivoli Event Services Manager to check authorizations. If a user has been deleted from the Tivoli Event Services Manager, that user needs to be deleted from the product repository also.

Keep in mind that having products implement self-managed security means that you have several security repositories spread over your system. All of these repositories need to be managed. A user's authorization scope within a system cannot be clearly defined unless all existing repositories are included in the analysis.

A good example for self-managed security is DB2. The protection of data within DB2 can be realized in two different ways:

- ▶ One way is to use the Tivoli Event Services Manager for data authorization checking, providing several general resource classes within RACF, that are used to decide which access authority is granted for a user within DB2. The DB2 objects (tables, packages, plans, and so on) are each protected by these distinct classes.
- ▶ The second way to implement DB2 security is to use the DB2 catalog for authorization checking. This is called DB2 *native* security. Native DB2 authorization uses the grant and revoke statements to keep the access privilege information in the DB2 catalog. Checking access means checking the DB2 catalog. DB2 determines what access rights are to be granted. This DB2 native security is independent from the Tivoli Event Services Manager.

1.2.2 External security manager

A common way of implementing security for software on z/OS is to use the Tivoli Event Services Manager. It is invoked by a software product by either using the System Authorization Facility (SAF) API or a set of callable services provided by RACF. In this book, we use RACF.

If software is relying on RACF as a security repository, there is a representation of functions and services that the software offers within RACF. It is common for software products to use installation-defined general resource classes in RACF. The representation of functions and services needs to adhere to RACF standards for profiles (as determined by the class descriptor table entry for that class).

With these profiles in place, software may issue a `RACROUTE REQUEST=AUTH` macro to the SAF API, which is then passed to RACF. The macro includes the requesting user's identity (the ACEE) and the resource name the user wants to access. The resource name is the RACF representation of the service from which the user requests authorization.

RACF then checks authorization for that resource name. A return code is calculated and sent back to the software. Based on the authorization of the user, RACF usually either suggests allowing the access (RC 0), denying it (RC 8), or indicates that it cannot determine the best choice (RC 4). The last is the most common case, although the default return code can be modified in the class descriptor table (CDT) entry of the class.

Clarification: RACF does not prevent access. The final decision of what access to grant is made by the requesting software (z/OS resource manager or installation software). RACF provides only “suggestions” to the requesting software, based on the information that is stored in the RACF database.

Keep in mind that software may also run in an authorized state. Program authorization is described in detail in the first volume of this series, *Reduce Risk and Improve Security on IBM Mainframes: Volume 1 Architecture and Platform Security*, SG24-7803. There are several things to consider for software that is authorized, including, for example, the possibility of bypassing RACF decisions when the software executes operating system services.

Figure 1-3 illustrates the authorization flow if software security is implemented in RACF. A user requests a service from a software, and then the software requests information from RACF. RACF delivers a return code back to the software, and the software allows or denies access to the function.

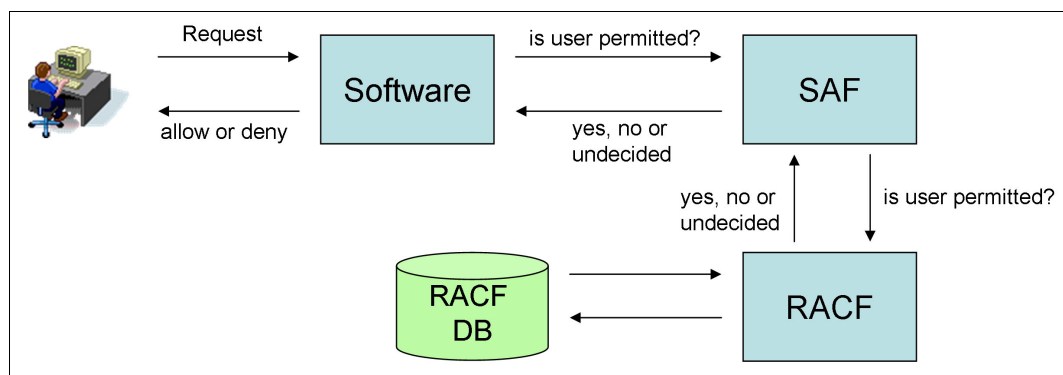


Figure 1-3 RACF decision flow

1.2.3 Exits

Some z/OS software products might provide *user exits*, also referred to as *installation exits*. A user exit is similar to a z/OS system exit. A software program can give control to an exit routine at defined points during execution, enabling the user to add code to the software and provide additional functionality. Software might, for example, provide an exit point, where an exit routine receives control and is able to manipulate input data.

The same concept can be used for security processing. Rather than using the Tivoli Event Services Manager or self-managed security by default, software can leave that decision up to the installation, based on what exit was chosen and defined to the exit point. For example, DB2 external security using RACF is implemented as an installation exit in DB2. The module receives control before authorization checking and, if active, provides user authorization decisions by RACF to DB2. If it is not active, DB2 internal security is used.

The same holds, for example, for the IBM Session Manager. It also provides an exit point for security processing. Depending on the exit routine that is defined for the exit point, authorization may be either self-managed or provided by the Tivoli Event Services Manager.

An exit might also be relevant to security because it receives control before authorization checking by the Tivoli Event Services Manager. When this is the case, the exit can potentially modify the identity information that is used by the program to be provided to the Tivoli Event Services Manager. Based on how the exit point is coded into the program, the exit might be able to modify important information that is used within the processing of the program, including authorization information.

1.2.4 Audit and compliance reporting

It might be difficult for a non-product specialist to understand the configuration parameters for the security-related setup of the middleware components that we discuss in this book. It might also be difficult for your security personnel to gather the needed information. Every one of these products is configured differently, so you face different concepts and approaches to how security is implemented.

Therefore, it is important to have a consolidated view of all of the security-related configuration for all middleware products. This enables nontechnical audit staff to better understand what the security setup is or all of the products without having to look into the products in detail and without having to rely on information that is sent from the product administration staff in your organization.

For most organizations, it is mandatory to perform security intelligence and compliance audits on their z/OS systems. The need for a comprehensive solution to fulfil those needs is evident in many customer situations, where, based on regulatory requirements, a compliance and audit solution for z/OS is almost always mandatory.

We use DB2, CICS, and IMS as an example. These products, or a subset of them, are found in most z/OS installations. For IBM clients, these may be the most important components of z/OS regarding business applications and processes. For IBM clients, using DB2 or IMS as the data backbone and CICS or IMS as transaction servers may be the most important components.

Although all of these products offer features for audit and compliance reporting themselves, IBM also offers a solution to fulfill these requirements for the products within the primary security software solution for z/OS, IBM Security zSecure.

The Audit component of Security zSecure offers functions to easily and independently perform audit and compliance reporting against it separate from the product itself. This component uses its own mechanisms to create a snapshot of the current system settings, including I/O information, system control blocks, authorized program information, UNIX data, and other information. This snapshot also includes configuration information for IMS, CICS and DB2, if available.

The collected information can be used for reporting on the security-related configuration of the product. The reporting is performed by using Security zSecure panels or the product's own language, CARLa. This format is known to the security and audit personnel if Security zSecure is deployed in an organization. Security staff gets the information that they need to audit the products and to be able to do that in a familiar environment.

It is also possible to create custom reports using CARLa to be able to report on configuration compliance and changes in the configuration. Continuous audit and customized security intelligence for the backbone middleware DB2, CICS, and IMS is achieved by using Security zSecure Audit.

The following reports are included in Security zSecure Audit:

- ▶ CICS region reports
- ▶ CICS transactions selection and reports
- ▶ CICS programs selection and reports
- ▶ IMS control region reports
- ▶ IMS transaction reports
- ▶ IMS program specification blocks
- ▶ DB2 Region overview and system privileges
- ▶ DB2 Sets of tables, indexes, and table spaces
- ▶ DB2 Sets of files comprising Java applications
- ▶ DB2 Packages (pre-bound SQL statements)
- ▶ DB2 Plans (control structures created during BIND)
- ▶ DB2 Sets of storage objects (volumes)
- ▶ DB2 Stored procedure and user function routines
- ▶ DB2 user-defined objects that define a numerical sequence
- ▶ DB2 Tables and views
- ▶ DB2 table spaces (data set name space for storing tables)

1.3 Considerations for programming, configuration, and performance

In implementation of mainframe software, the design and programming of an application and the configuration of it directly influence the security of the system. These points, combined, have an impact on the performance and the usability of the application.

An application, which is primarily designed to goals of performance and usability, must compromise on security simply because a strict security design of an application is likely to have an impact on the usability and the performance. The impact on usability might be, for example, that a user must enter a password at several points within the application. A performance impact of security can be extensive logging of the application or the use of cryptography.

If the configuration of an application follows strict security standards, this leads to the same implications as stated previously. If you configure your applications to be as secure as possible, you might have a negative impact on usability and performance. However, if you configure your application with no security at all, your users will like the application, because they do not have to enter their passwords, and that is fast. But anyone will be able to compromise your data.

The programming, configuration, and security implementation of a product need to be in a well-balanced state. Although there is no perfect security system, you can implement strong security without sacrificing other goals. That has an impact on several other goals that you might have.

Also, keep in mind that the security of your infrastructure is largely dependant on the security of your applications. An attacker from the outside will probably try to use an applicaiton interface to enter the system. However, these interfaces are protected by the application's security measures, not the operating system. So although you might have implemented strong operating system security, as we advise you to do, your system might still be vulnerable because your application interfaces with the outside world provide attack vectors.

1.4 Logging

In this section, we describe common approaches to logging and recording of events for products used on z/OS.

Logging in an information technology context means writing a history of actions and changes. It is the recording of data about specific events and is vital to problem determination, auditing, accountability and system access reporting. This recorded data is maintained in a data file, called a log, for later investigation and possible analysis. The log should contain information about valid accesses and about attempted compromises of system security controls. Some security products and tools examine and format detailed reports that can be used to present facts on selective actions taken on z Systems. On z/OS, there is a central logging facility for the operating system and for applications and subsystems. This is the Systems Management Facility (SMF). In addition to SMF, many products may also use internal logging and write logs data sets.

A log is important to security. Most logs contain identity information of the requesting entity for any specific log event. Therefore, violations and exceptions and the use of authorities can be reconstructed using log information. It is especially important to archive log information for an appropriate amount of time. This can help to reconstruct security events that happened in the past. The protection of log data sets is also important, as a write permission on log data can be used by an attacker to cover his actions.

1.4.1 Internal logging and the syslog

The collection of log information without external logging infrastructure, such as SMF or the syslog daemon in UNIX System Services, is used by some products on z/OS. Most z/OS products use of the existing log infrastructure.

Internal logging makes sense only for certain products, such as data management-related software. For example, DB2 uses the logging infrastructure with SMF but also provides archive logs that contain the data management operations performed by DB2. This information is not suitable to be stored in SMF; therefore, DB2 manages this data. However, IMS does not use SMF or syslog processing. It relies solely on special-purpose IMS log data sets. During IMS execution, all relevant log data and information necessary to restart the system if of hardware or software failure is recorded in a system log data set.

If products use internal logging, they probably write log information to special-purpose data sets. These data sets require the same attention from the security personnel as SMF data, probably even business data. They may contain important information about the processing of the product. Also, some logs might contain configuration-related information. Especially log data sets of database managers do contain the data that is written to the database or deleted from it.

In addition to internal logging and SMF, many products also use the system log to provide information to the user. If the system log is used by a product, it is used to issue informational messages, but more importantly warning, and error messages to the administrator or the operator. For example, a security violation in a product is almost always directed to the system log. Messages on the system log are often used by automation and monitoring software. Automation tools, for example, are triggered by certain message IDs on the system log and then perform tasks that are defined for that message within system automation.

1.4.2 SMF

System management facility (SMF) records log data from the operating system and applications either to special purpose data sets or coupling facility structures (so-called *log streams*). System and application-related log information your installation can be used for the following purposes:

- ▶ Billing users
- ▶ Reporting reliability
- ▶ Analyzing the configuration
- ▶ Scheduling jobs
- ▶ Summarizing direct-access volume activity
- ▶ Evaluating data set activity
- ▶ Profiling system resource use
- ▶ Maintaining and auditing system security

The data collection is executed by several specific routines spread throughout z/OS and other products.

The volume and variety of information in the SMF records enables installations to produce detailed analysis and summary reports. For example, by keeping historical SMF data and studying its trends, an installation can evaluate changes in the configuration, workload, or job scheduling procedures. Similarly, an installation can use SMF data to determine system resources wasted because of problems, such as inefficient operational procedures or programming conventions. Also, archived SMF records provide security personnel with the information to reconstruct and investigate past security incidents.

The primary benefit of using SMF for logging is its heavy use throughout z/OS and its components. All production environments have SMF processing functions in place. SMF is a single point of entry for logging purposes. All information that may be needed for event reporting is included in the SMF records from all kinds of sources. An application that is choosing to use SMF logging is automatically included in the SMF processing in a production environment. SMF records are archived once for all participating functions, and there are fewer sources for logging if you use SMF whenever possible.

Many of the products described in this book use SMF for logging. An application that will use SMF must issue the SMF assembler macro in an authorized state. As stated previously, SMF records typically include security and audit-related events. Based on these events, a product-specific security reporting can be implemented.

SMF records are usually archived for a long time in a productive environment. This enables the security analyst to use SMF to re-create events that led to incidents even if the incident has happened some time ago. Based on the identity information included in the SMF record for a certain event, it can be related to RACF SMF records.

This is useful to understand if a security incident has happened because a user received unusually high privileges in the system. Based on the SMF records of the product and the RACF SMF records, you can reconstruct the events that led to the security incident not only from an application point of view but also from the RACF perspective.

Figure 1-4 illustrates the functional overview of the SMF process and the list that follows describes each step.

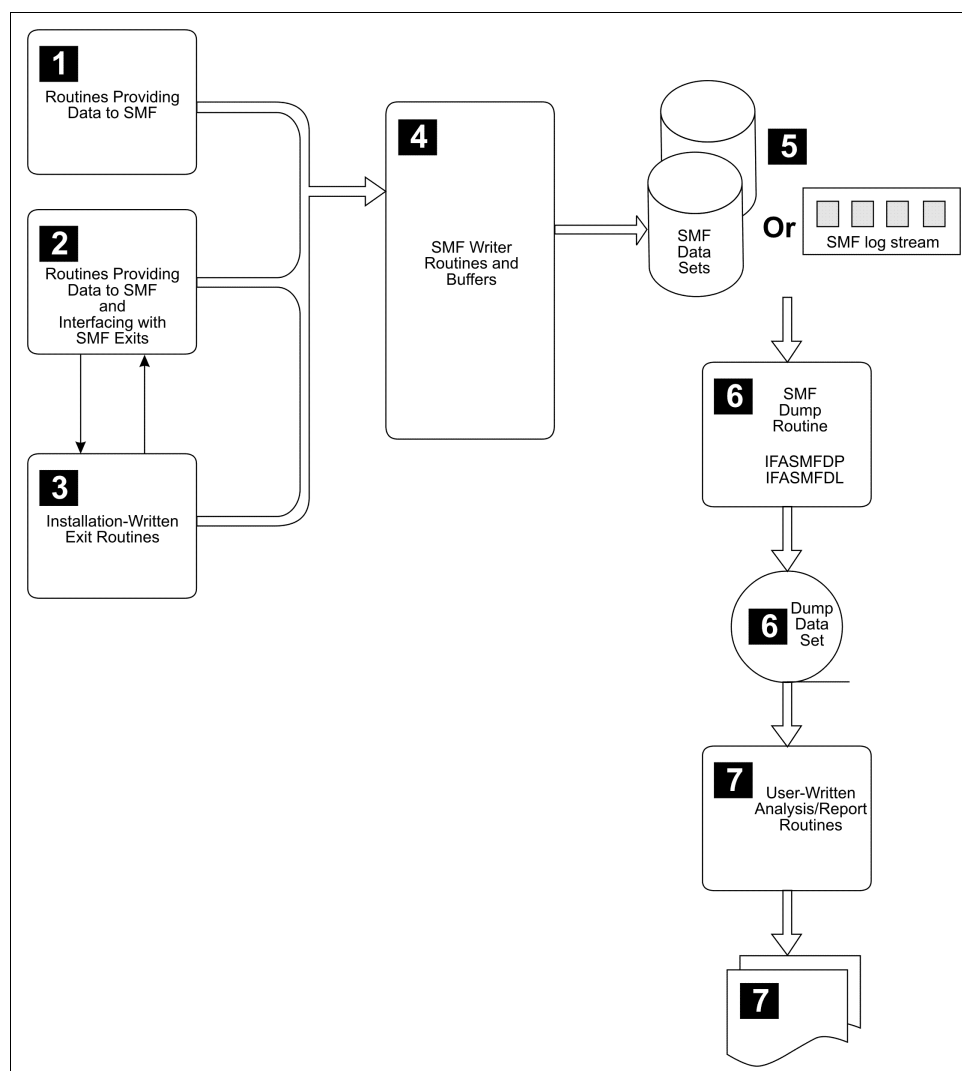


Figure 1-4 The SMF process illustrated

1. Routines (SMF, system, licensed program, installation-written) collect and format data into records and then pass the records to the SMF writer.
2. In addition to collecting data for SMF, some routines interface with the SMF exits, passing control to them at several points during job (and job step) processing.
3. SMF routines copy records to SMF buffers and then transfer records from the SMF buffers to either of these locations:
 - SMF data sets
 - SMF log streams
4. SMF routines then issue messages to the operator indicating the successful or unsuccessful completion of specific SMF-related events.

5. Data is written to the SMF data store:
 - If you write your SMF data to SMF data sets, the data sets are filled one at a time. While SMF writes records on one data set, other SMF data sets can be dumped or cleared.
 - If you write your SMF data to a log stream, you can keep writing data to your log stream. The log stream offloads to DASD data sets when the log stream coupling facility structure (or the local storage buffers for a DASD only log stream) fills. You define the thresholds in the log stream definition for how much data is held before offload.
6. The SMF dump programs copy data from either SMF data sets or log streams to tape or direct-access data sets for permanent storage. You can also use the SMF log stream dump program to dump to a temporary data set for immediate use.
7. Analysis and report routines, either user-written or those such as the Tivoli Decision Support for z/OS licensed program, process information records.



Database managers

In this chapter, we cover security concepts and provide an architectural overview of database managers on IBM z Systems. We also explain guiding principles for configuring database security.

This chapters covers the following database management systems:

- ▶ IBM DB2 for z/OS
- ▶ IBM Information Management System
- ▶ Virtual Storage Access Method

2.1 IBM DB2 for z/OS

DB2 for z/OS Version 11 provides critical enhancements to security and auditing. These enhancements strengthen DB2 security in the z/OS environment.

Certain privileges and authorities are assigned when you install DB2 for z/OS. You can reassign these authorities by changing the DSNZPARM subsystem parameter. Other privileges and authorities are managed through authorization IDs and roles.

In this section, we discuss securing data stored in the DB2 database management system.

2.1.1 Security concepts and architecture

The list of potential security problems that can be encountered when storing data, both sensitive and non-sensitive, is by no means exhaustive. Over the years, DB2 developers have recognized and addressed the following security problems:

- ▶ Privilege theft or mismanagement
- ▶ Application or application server tampering
- ▶ Data or log tampering
- ▶ Storage media theft
- ▶ Unauthorized access to objects

To address these security problems, DB2 provides the following security capabilities, which are covered in the topics that follow:

Authentication	Verifies that the user allowed to use this system
Authorization	Provides and controls the correct level of authorization when a user accesses the system
Data integrity and confidentiality	Ensures that data is not illegally modified or corrupted and keeps confidential data private or restricted
Audit	Traces access to data
System integrity	Ensures that the system has not been compromised

Authentication

There are two modes of security access control to the DB2 database system today. *Authentication* to DB2 is managed by facilities that reside outside the DB2 system, whereas *authorization* is managed by the database manager.

Authentication is the first security capability encountered when a user attempts to use the DB2 for z/OS product. The user must be identified and authenticated before being allowed access to the data, which is handled by the authorization capability.

Authentication is the process by which a system verifies a user's identity. User authentication is completed by a security facility outside the DB2 database system. The Access Control Authorization Exit (DSNX@XAC) enables you to use external security such as Resource Access Control Facility (RACF) for authorization checking for DB2 objects, authorities, commands, and utilities. There are certain instances where the authorization checking done by RACF is different from the authorization checking done by DB2.

When DB2 authentication is done via RACF, DB2 obtains RACF information through the DSNX@XAC exit. DB2 then caches the successful authorization for execution of packages, routines, and dynamic statements.

The authentication process produces a DB2 authorization ID. Group membership information for the user is also acquired during authentication. Default acquisition of group information relies on DSNX@XAC that is included when you install the DB2 database management system. If you prefer, you can acquire group membership information by using a specific group-membership plug-in module, such as lightweight directory access protocol (LDAP).

Starting from DB2 for z/OS V10, DB2 is enabled to support z/OS client login through the use of digital certificates. Previously, RACF users of traditional, remote client applications authenticated themselves to DB2 by providing their user IDs and passwords. Now, by using z/OS digital certificates, the Secure Sockets Layer (SSL) protocol supports server and client authentication during the handshake phase. This enhancement enables a server to validate the certificates of a client at the server, preventing the client from obtaining a secure connection without an installation-approved certificate. The authentication of the remote client's digital certificate is performed for DB2 by Application Transparent Transport Layer Security (AT-TLS) that is provided with the z/OS Communications Server TCP/IP stack. Support for z/OS client login by using digital certificates also provides the benefit of RACF certificate name filtering. A certificate name filter enables you to associate many client certificates with one user ID based on the unique user information in the certificate, such as the user's affiliation. You can create one or more certificate name filters to map a large number of client certificates to a limited number of user IDs, which helps you reduce administrative costs.

Additionally, DB2 for z/OS V11 addresses the following during authentication when using an external security manager:

- ▶ The **OWNER** keyword is honored to control authorization when using the DSNX@XAC exit authorization:
 - Support of the **OWNER** keyword for the **BIND** and **REBIND** commands.
 - Support owner authorization during **AUTOBIND**.
 - Support **DYNAMICRULES(BIND)** behavior for dynamic SQL statements.
- ▶ Refresh DB2 cache entries when RACF permissions change. The following DB2 cache entries can be refreshed when the access control authorization exit is active and RACF permissions change:
 - Package
 - Authorization cache
 - Routine authorization cache
 - Dynamic statement cache
- ▶ RACF access control module (DSNXRXAC) support:

To support new functionality, DB2 11 introduces the following changes in the RACF access control module (**DSNXRXAC**):

 - Support the Global Variable privileges, **READ (READAUTH)** and **WRITE (WRITEAUTH)**. IBM supplied RACF resource class for global variable is **MDSNGV/GDSNGV**.
 - Return the **RACLSTED** classes at DB2 start in the new **XAPL** field, **XAPLCLST**.
 - Support all authorization checks that are associated with **AUTOBIND** requests for user-defined functions. This removes the return code 8 and reason code 17 issued for the authorization failures associated with **AUTOBIND** requests for user-defined functions.

Authorization

After a user is authenticated, the database manager determines which, if any, DB2 data or resources a user is allowed to access. DB2 controls access to its objects and data through authorization identifiers (IDs) and roles and the privileges that are assigned to them. Each privilege and its associated authorities enables a user to take specific actions on an object.

DB2 processes are represented by a set of identifiers (IDs). There are three types of identifiers: primary authorization IDs, secondary authorization IDs, and SQL IDs.

Primary authorization ID	Those granted to the authorization ID directly.
Secondary authorization ID	Those granted to the groups and roles in which the authorization ID is a member.
SQL authorization ID	(SQL ID) holds the privileges that are exercised when a process issues certain dynamic SQL statements.

Authorization can be granted to users in the following categories:

- ▶ **System-level authorization**
The system administrator (SYSADM), system control (SYSCTRL), and system maintenance (DBMAINT) authorities provide varying degrees of control over instance-level functions. Authorities provide a way both to group privileges and to control maintenance and utility operations for instances, databases, and database objects.
- ▶ **Database-level authorization**
The security administrator (SECADM) and database administrator (DBADM) authorities provide control within the database. Other database authorities include LOAD (ability to load data into a table) and CONNECT (ability to connect to a database).
- ▶ **Object-level authorization**
Object-level authorization involves checking privileges when an operation is performed on an object. For example, to select from a table a user must have SELECT privilege on a table (as a minimum).
- ▶ **Content-based authorization**
Views provide a way to control which columns or rows of a table that specific users can read.

You can use these features, in conjunction with the DB2 audit facility for monitoring access, to define and manage the level of security your database installation requires.

When an application gains access to a subsystem, the user has been authenticated and access to DB2 for z/OS is checked using RACF. DB2 for z/OS controls access to data through authorization IDs or roles. DB2 relies on IDs or roles to determine whether to allow or prohibit certain processes. DB2 assigns privileges and authorities to IDs or roles so that the owning users can take actions on objects.

Data can be accessed from a batch program, from a user on an interactive terminal session, or from a CICS or IMS transaction. For the purposes of security, DB2 uses the term *Process* to represent all forms of access to data as shown in Figure 2-1.

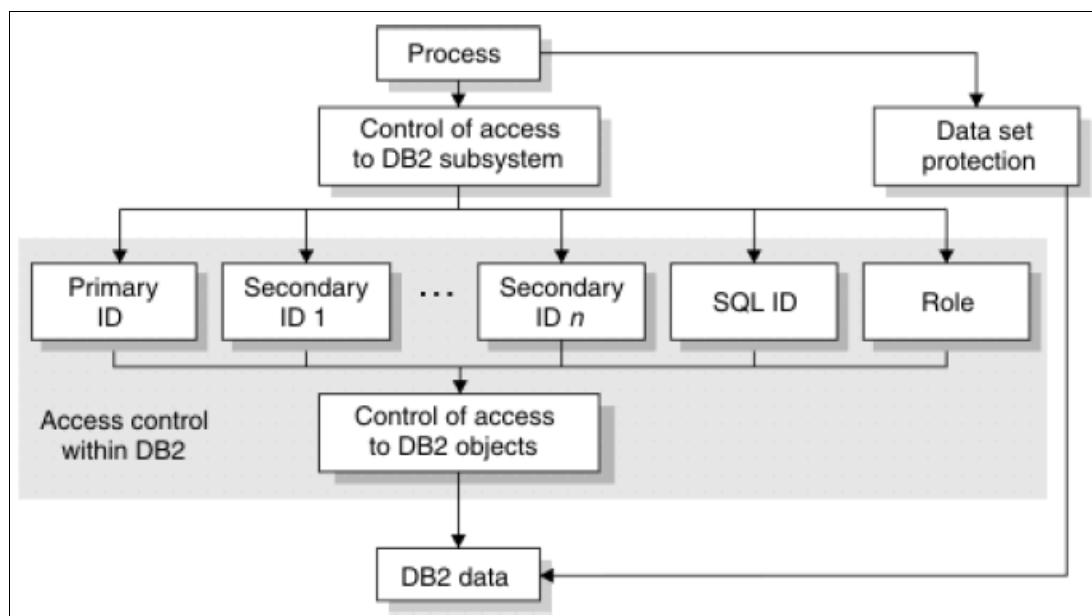


Figure 2-1 Access control process

Access control within DB2, uses identifiers (IDs) to control access to DB2 objects. The process must first satisfy the security requirements to access the DB2 subsystem. When the process is within the DB2 subsystem, DB2 checks for a primary authorization ID, secondary authorization ID, or SQL ID to determine whether the process can access DB2 objects. If the process has the necessary ID or IDs, it can access DB2 objects, including DB2 data.

Data integrity and confidentiality

Data integrity involves the assurance that the data being accessed or read has not been tampered, altered, nor damaged since the last authorized access. Integrity involves maintaining and assuring the accuracy and consistency of data through its lifecycle.

Data integrity controls are the basis for every database management system, with the ACID properties (atomicity, consistency, isolation, and durability).

Confidentiality is the set of rules or a promise that limits access or places restrictions on information. It is basically equivalent to privacy. Confidentiality involves protecting data from unauthorized access or disclosure.

DB2 makes use of referential integrity which ensures data integrity by enforcing rules with referential constraints, check constraints, and triggers. You can rely on constraints and triggers to ensure the integrity and validity of your data, rather than relying on individual applications to do that work.

DB2 key propagation helps preserve the referential integrity of the transformed data across applications, databases, and operating environments.

Data integrity also means preventing any corruption or loss of data. To prevent corruption of data we need to ensure proper encryption of the data. Encryption protects against unauthorized access to data and provides the highest level of data security possible.

There are many ways to handle encrypted data within DB2:

- ▶ DB2 built-in-function support for encryption
- ▶ IBM InfoSphere® Guardium® Data Encryption for DB2 and IMS databases. It uses IBM z Systems cryptographic hardware to protect sensitive data at the DB2 row level and IMS segment level
- ▶ Disk storage-based encryption with IBM System Storage® DS8000®
- ▶ Tape storage encryption
- ▶ SSL protocol through the z/OS Communications Server IP Application Transparent Transport Layer (AT-TLS) service
- ▶ IBM Encryption Facility for z/OS
- ▶ Advanced Encryption Standard (AES) for encrypting user IDs and passwords over network connections
- ▶ DB2 edit procedures or field procedures, which can use the Integrated Cryptographic Service Facility (ICSF)
- ▶ IBM Data Encryption for IMS and DB2 Databases tool

For more information about using data encryption with DB2, see Chapter 6, “Cryptography for DB2 data,” in the IBM Redbooks publication titled *Security Functions of IBM DB2 10 for z/OS*, SG24-7959.

To address loss of data we need to provide capabilities for backup and recovery. DB2 supports recovering data to its current state or to an earlier state. You can recover table spaces, indexes, index spaces, partitions, data sets, and the entire system. You can also compare database images to validate changes and preserve data integrity.

Disaster recovery considerations

When planning security for a disaster recovery site exercise in the following sections, take these necessary actions into account:

- ▶ Hardware symmetry

Ensure that the same type of hardware and operating system support is available at the recovery site. This is important in an environment where you have one of CRYPTO EXPRESS COPROCESSORS (CEXnC or CEXnA) encrypted CKDS and require API support delivered through the use of CRYPTO EXPRESS COPROCESSORS enabled services. This includes the use of clear key encryption in ICSF environments before HCR7751, and any use of the Secure Key API.
- ▶ Recovery assets

CKDS contains all of the generated data encrypting keys and the DES Master Key verification pattern used to validate the hardware registers at ICSF startup. As such, consider the CKDS to be a system-critical file, and back it up with the same frequency as other critical z/OS system data sets, such as SYS1.PARMLIB.

Many customers place these critical data sets on a special volume.
- ▶ Master Key Entry at recovery site

At the disaster recovery site, there needs to be a designated key officer who can run the ICSF Master Key entry dialog to enter the same master key used to initialize the CKDS.
- ▶ Secure the disaster site

The disaster recovery site must be in a safe area and easily accessible

Audit

Security auditing allows you to inspect and examine the adequacy and effectiveness of the policies and procedures that you put in place to secure your data. DB2 provides the ability for you to monitor if your security plan is adequately designed based on your security objectives and determine whether your implementation techniques and procedures are effectively carried out to protect your data access and consistency. It enables you to address the following fundamental questions about your data security:

- ▶ What sensitive data requires authorized access?
- ▶ Who is privileged to access the data?
- ▶ Who has actually accessed the data?
- ▶ What attempts are made to gain unauthorized access?

There are many kinds of audit information available in DB2. The DB2 catalog stores the definitions of all the objects and corresponding authorizations. The DB2 recovery log and utilities or tools are also helpful in finding out how and when data was modified.

Security auditing allows you to inspect and examine the adequacy and effectiveness of the policies and procedures that you put in place to secure your data.

The DB2 catalog contains critical authorization and authentication information. This information provides the primary audit trail for the DB2 subsystem. You can retrieve the information from the catalog tables by issuing SQL queries. Most of the catalog tables describe the DB2 objects, such as tables, views, table spaces, packages, and plans. Other tables, particularly those with the AUTH character string in their names, hold records of every granted privilege and authority. Each catalog record of a grant contains the following information:

- ▶ Name of the object
- ▶ Type of privilege
- ▶ IDs that receive the privilege
- ▶ IDs that grant the privilege
- ▶ Time of the grant

The DB2 audit trace can help you monitor and track all the accesses to your protected data. The audit trace records provide another important trail for the DB2 subsystem. You can use the audit trace to record the following access information:

- ▶ Changes in authorization IDs
- ▶ Changes to the structure of data, such as dropping a table
- ▶ Changes to data values, such as updating or inserting records
- ▶ Access attempts by unauthorized IDs
- ▶ Results of GRANT statements and REVOKE statements
- ▶ Mapping of Kerberos security tickets to IDs
- ▶ Other activities that are of interest to auditors

In addition to DB2 auditing and log capabilities, every organization should investigate using an external *data activity monitoring* solution that can collect deep-scale auditing information at a network level. That can help eliminate performance or other effects on database operations.

IBM offers the IBM InfoSphere Guardium family of products. Data collection InfoSphere Guardium can collect and correlate many different types of information into an administration repository:

- ▶ Modifications to an object (SQL UPDATE, INSERT, DELETE)
- ▶ Reads of an object (SQL SELECT)

- ▶ Explicit GRANT and REVOKE operations (to capture events where users might be attempting to modify authorization levels)
- ▶ Assignment or modification of an authorization ID
- ▶ Authorization attempts that are denied due to inadequate authorization
- ▶ CREATE, ALTER, and DROP operations against an object (such as a table)
- ▶ Utility access to an object (IBM utilities only)
- ▶ DB2 commands entered (including the ability to determine which users are issuing specific commands)

For more information, see “DB2 security” in the IBM Knowledge Center:

<http://ibm.co/1FLFMSI>

For more information, see “InfoSphere Guardium S-TAP for DB2 on z/OS V9.0 documentation” in the IBM Knowledge Center:

<http://ibm.co/1Ln0YAJ>

System integrity

The z Systems and z/OS commitment to system integrity means that unauthorized users and programs cannot bypass the hardware isolation functions that protect other users or programs, cannot obtain control in an authorized execution status, and cannot bypass the system-level security functions provided by z/OS Security Server.

IBM z Systems provide different kinds of hardware isolation functions that create a strong foundation for security.

Basic isolation functions provide mandatory separation of users and applications from each other and from the system such as:

- ▶ Storage protection keys

The hardware provides 16 protection keys that the system can assign to running programs and to areas of storage with options that can require that a program's key match the storage area's key before the program can write into the storage, or optionally, before the program can even read from the storage.

- ▶ Multiple address spaces

The hardware and software isolate user (and system) programs into different *address spaces*. Each address space has the ability to read common system storage, but it cannot read nor write the non-shared storage that belongs to another address space unless allowed to do so by an authorized program.

In this section, we describe the following topics:

- ▶ Security setup and preparation
- ▶ Adapters and interfaces
- ▶ Data sharing and parallel sysplex
- ▶ Event monitoring and recording

Security setup and preparation

With key enhancements in DB2 V11, DB2 for z/OS and z Systems continue to lead the industry in security and auditing. With DB2 V11 RACF Exit enhancements, external security managed by RACF administrators can now fully handle access to DB2 objects.

- ▶ The use of the **OWNER** keyword is now acceptable by RACF. A new installation parameter allows the use of the package **OWNER** for static and dynamic SQL authorization.
- ▶ A refresh function for DB2's authorization cache has been implemented allowing RACF to dynamically notify changes and keep the security definitions synchronized.
- ▶ DB2 11 also removes some of the restrictions on the use of the **SQL GROUP BY**, **DISTINCT**, and **UNION** clauses when querying a masked table.

DB2 uses different levels of authority and grantable privileges that must be managed with care to properly secure access to database resources. In the next sections we present the different authority levels that are needed in DB2 to manage database resources.

Administrative authority level that operates at the instance level

There is only one administrative privilege level that can operate at the instance level, the **SYSADM**:

▶ **SYSADM**

The **SYSADM** authority includes all the privileges, including system privileges, for creating objects and accessing all data. With the **SYSADM** authority, an authorization ID or role can perform the following actions and grant other IDs the required privileges to perform them:

- Use all the privileges of **DBADM** over any database
- Use **EXECUTE** privileges on all packages
- Use **EXECUTE** privileges on all routines
- Use **USAGE** privilege on distinct types, JARs, and sequences
- Use **BIND** on any plan and **COPY** on any package
- Use privileges over views that are owned by others
- Set the current SQL ID to any valid value
- Create and drop synonyms and views for other IDs on any table
- Use any valid value for **OWNER** in **BIND** or **REBIND**
- Drop database DSNDB07

An authorization ID or role with **SYSADM** authority can also perform the following actions, but cannot grant other IDs the privileges to perform them:

- Drop or alter any DB2 object, except system databases
- Issue a **COMMENT ON** statement for any table, view, index, column, package, plan
- Issue a **LABEL ON** statement for any table or view
- Terminate any utility job

Administrative authority levels that operate at the database level

There are two administrative privilege levels that can operate at the database level, **DBADM** and **SECADM**:

► **DBADM**

DBADM has the following privileges on all tables in a database:

- **ALTER**
- **DELETE**
- **INDEX**
- **INSERT**
- **REFERENCES**
- **SELECT**
- **TRIGGER**
- **UPDATE**

DBADM also includes authorities of **DBCTRL** and **DBMAINT**, which are shown in “System control authority levels that operate at the instance level” on page 32.

► **SECADM**

SECADM authority enables you to manage security-related objects in DB2 and control access to all database resources. It does not have any inherent privilege to access data stored in the objects, such as tables. With the **SECADM** authority, you can perform the following tasks:

- Create, alter, drop, and comment on row permissions
- Create, alter, drop, and comment on column masks
- Activate and deactivate row access control
- Activate and deactivate column access control
- Create, drop, and comment on roles
- Create, alter, drop, and comment on trusted contexts
- Create and comment on secure triggers and user-defined functions
- Alter the **SECURED** or **NOT SECURED** clause on triggers and user-defined functions
- Create audit policies by inserting rows into the **SYSIBM.SYSAUDITPOLICIES** catalog table
- Access and update the **SYSIBM.SYSAUDITPOLICIES** catalog table which records audit policy definitions
- Has implicit **SELECT** access on all catalog tables and implicit **INSERT**, **DELETE**, and **UPDATE** privileges on updatable catalog tables
- Grant and revoke all grantable privileges and authorities
- Issue the **TRACE** command to start, stop, and display a trace

System control authority levels that operate at the instance level

There are three system control privilege levels that can operate at the database level: **SYSCTRL**, **DBCTRL**, and **DBMAINT**.

► **SYSCTRL** (system control)

SYSCTRL authority level provides control over operations that affect system resources. For example, a user with **SYSCTRL** authority can create, update, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data.

► **DBCTRL**

DBCTRL has the following privileges on a database:

- **DROP**
- **LOAD**
- **RECOVERDB**
- **REORG**
- **REPAIR**

► **DBMAINT** (system maintenance)

DBMAINT provides the authority required to perform maintenance operations on all databases associated with an instance. A user with **DBMAINT** authority can update the database configuration, back up a database or table space, restore an existing database, and monitor a database. Like **SYSCTRL**, **DBMAINT** has the following privileges on a database:

- **CREATETAB**
- **CREATETS**
- **DISPLAYDB**
- **IMAGCOPY**
- **STATS**
- **STARTDB**
- **STOPDB**

Differences in system privileges for different versions of DB2

Because many organization today still use DB2 versions older than V11 we thought it is a prudent idea to list some of the differences inherent to these versions.

The left column in Table 2-1 shows some of the security-related privilege levels that are available in versions of DB2 before V10, and the right column shows those that were added with V10.

Table 2-1 System authorities and privileges for security

Before DB version 10	New in DB2 version 10 and later
SYSADM	SYSADM includes these authorities: SECADM, SYSTEM DBADM, SQLADM. ACCESSCTRL, DATAACCESS. SYSADM also includes the new EXPLAIN and CREATE_SECURE_OBJECT privileges.
DBADM	DBADM Included authorities: DBCTRL, DBMAINT
DBCTRL	DBCTRL Included authorities: DBMAINT
SYSCTRL	SYSCTRL Included authorities: Installation SYSOPR, DBCTRL, DBMAINT, and ACCESSCTRL

- The **ACCESSCTRL** authority allows you to grant explicit privileges to authorization IDs or roles by issuing **SQL GRANT** statements. It enables you to grant privileges on all objects and resources, except the **CREATE_SECURE_OBJECT** privilege and the system **DBADM**, **DATAACCESS**, and **ACCESSCTRL** authorities.
- The **DATAACCESS** authority allows you to access and update data in user tables, views, and materialized query tables in a DB2 subsystem. It also allows you to execute plans, packages, functions, and procedures.

- The **SQLADM** authority allows you to issue the **SQL EXPLAIN** statements, execute the **PROFILE** commands, run the **RUNSTATS** and **MODIFY STATISTICS** utilities on all user databases, and execute system-defined routines, such as stored procedures or functions, and any packages that are executed within the routines.

Additional security enhancements in DB2 V11

DB2 V11 introduced the following new capabilities related to security.

- DB2 enhancements for exit authorization checking

DB2 provides the **ACEE** (access control environment element) of the package owner for authorization checking when the access control authorization exit is active.

The Access Control Authorization Exit (**DSNX@XAC**) enables use of external security, such as **RACF**, for authorization checking for DB2 objects, authorities, commands, and utilities. There are certain instances where the authorization checking done by **RACF** is different from the authorization checking done by DB2.

DB2 also refreshes the cache entries of the package authorization, the routine authorization, the **DDF** user authorization, and the dynamic statement when a user profile or resource access is changed in **RACF** and the access control authorization exit is active.

DB2 V11 refreshes DB2 cache entries when **RACF** permissions change. Previous to DB2 V11, when DB2 caches were enabled and **RACF** permissions changed in **RACF**, then the package authorization cache, routine authorization cache, and dynamic statement cache were not refreshed to reflect the change. To refresh the cache entries, **SQL GRANT** and **REVOKE** statements had to be issued or, to invalidate the entry from dynamic statement cache, the **RUNSTATS** utility had to be executed.

In DB2 for z/OS V11, the **DSNX@XAC** exit authorization is enhanced to:

- Support **OWNER** keyword for **BIND** and **REBIND** commands.
- Support owner authorization during autobind.
- Support **DYNAMICRULES(BIND)** behavior for dynamic SQL statements.

DB2 V11 introduces the capability to refresh the DB2 cache entries when the access control authorization exit is active and **RACF** permissions change. The following caches are refreshed:

- Package
- Authorization
- Routine authorization
- Dynamic statement

The following **RACF** Event Notifications (**ENF**) signals are used by DB2 V11 to refresh cache entries:

- **ENF 71** When a user's permission is changed in **RACF**
- **ENF 79** When a user's permission to access a resource is changed in **RACF**.
- **ENF 62** When **RACF** options are refreshed.

Figure 2-2 shows the relationship between DB2 V11 and RACF. Changes made in RACF (number 1 in the figure) are communicated to DB2 to take action accordingly (number 2 in the figure).

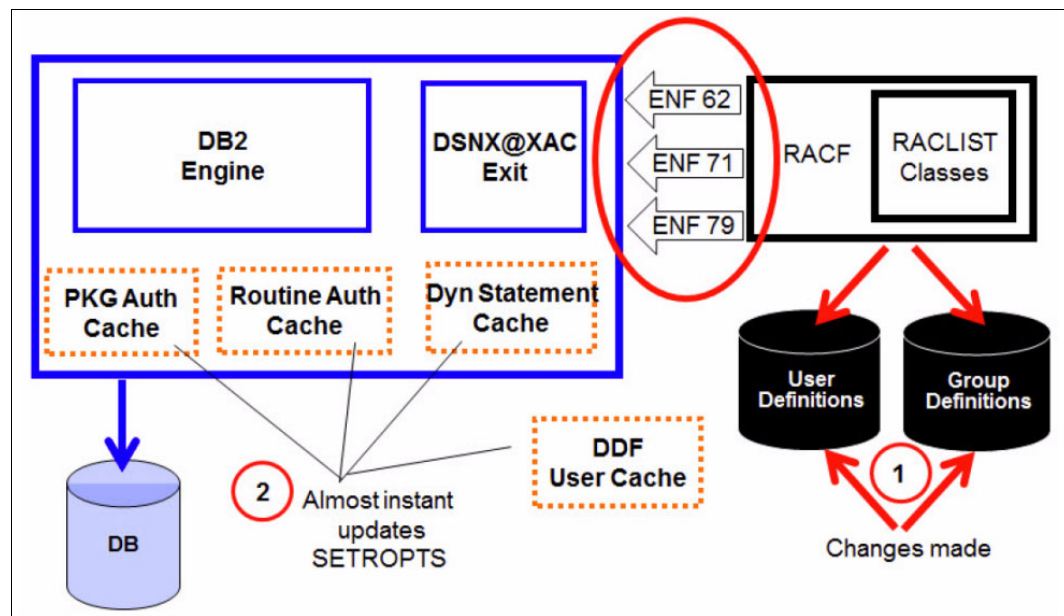


Figure 2-2 DB2 11 and RACF Access Control Authorization Exit Authorization

ENF 71 is issued for change in a user or group profile. When ENF 71 is issued, DB2 refreshes the cache entries for the affected user or group.

ENF 79 is issued for change in a user's or group's authorization to resources. When ENF 79 is issued, DB2 caches the resource changes, because the resource changes do not take effect until the **SETROPTS RACLIST REFRESH** command is issued.

ENF 62 is issued for the **SETROPTS RACLIST REFRESH** command. When ENF 62 is issued, DB2 refreshes the cache entries for the resources that are cached during ENF 79 notification.

DB2 V11 listens to the type 71 ENF signal issued by RACF for the following RACF commands:

- ALTUSER
- CONNECT
- DELUSER
- DELGROUP
- REMOVE

► DB2 enhancements for program authorization

DB2 provides the capability to check whether an application program is authorized to use a plan.

DB2 V11 provides an approach to verify that an application program using a DB2 application plan is correct when accessing DB2. A DB2 plan relates an application process to a local instance of DB2 and specifies processing options. One of the options is a list of package names that can be used by the application plan.

This list controls what packages can be used by any program that uses the plan. It is provided by the owner of the plan when the plan is bound to DB2. If any user is granted execute privileges on the plan, the user can execute any program using the plan and any package identified in the package list.

► DB2 enhancement to the masking functions

DB2 removes some restrictions related to aggregation of data while using the masking functions.

Row and column access control enables you to manage access to a table at the level of a row, a column, or both. You can implement row access control through row permissions and column access control through column masks.

A column mask is a database object that describes a specific column access control rule for a column. In the form of an SQL CASE expression, the rule specifies the condition under which a user, group, or role can receive the masked values that are returned for a column.

Network components

DB2 for z/OS V11 is often times used as a backend database server. Requesters can come from multiple front-ends using different technologies.

The Distributed Database Facility (DDF) is the DB2 on z/OS component that supports connectivity with other databases across the network. It implements the full distributed relational database architecture (IBM DRDA®) Application Requester / Application Server function types.

The distributed data facility (DDF) component allows client applications that run in an environment that supports DRDA to access data at DB2 servers. In addition, a DB2 application can access data at other DB2 servers and at remote relational database systems that support DRDA.

DDF supports TCP/IP and Systems Network Architecture (SNA) network protocols. DDF allows the DB2 server to act as a gateway for remote clients and servers. A DB2 server can forward requests on behalf of remote clients to other remote servers regardless of whether the requested data is on the DB2 server.

The remote site must have the proper security configured to access DB2 through DDF.

Figure 2-3 on page 37 is an example of the entries on the DDF panel. Note that the EXTENDED SECURITY field (EXTSEC) = Yes.

The EXTSEC subsystem parameter specifies how two related security options are to be set. These settings control what happens when a DDF connection has security errors and whether RACF users can change their passwords through the DRDA change password function.

When this parameter is set to YES, detailed reason codes are returned to the client when a DDF connection request fails because of security errors that might enable more malicious attacks. If this parameter is set to YES, RACF users can change their passwords by using the DRDA change password function.


```

DSNTIPR INSTALL DB2 - DISTRIBUTED DATA FACILITY PANEL 1
====> _
DSNT512I WARNING: ENTER UNIQUE NAMES FOR LUNAME AND LOCATION NAME
Enter data below:
1 DDF STARTUP OPTION      ===> NO          NO, AUTO, or COMMAND
2 DB2 LOCATION NAME      ===> LOC1        The name other DB2s use to refer to this DB2
3 DB2 NETWORK LUNAME     ===> LU1         The name VTAM uses to refer to this DB2
4 DB2 NETWORK PASSWORD   ===>            Password for DB2's VTAM application
5 RLST ACCESS ERROR      ===> NOLIMIT     NOLIMIT, NORUN, or 1-5000000
6 RESYNC INTERVAL        ===> 2          Minutes between resynchronization period
7 DDF THREADS            ===> INACTIVE    Status of a qualifying database access thread after
                                         commit. ACTIVE or INACTIVE.
8 MAX INACTIVE DBATS      ===> 0          Max number of type 1 inactive threads.
9 DB2 GENERIC LUNAME     ===>            VTAM LU name for this DB2 subsystem or data
                                         sharing group.
10 IDLE THREAD TIMEOUT   ===> 120        0 or seconds until dormant server ACTIVE thread
                                         will be terminated (0-9999)
11 EXTENDED SECURITY     ===> YES         Allow change password and descriptive security
                                         error codes. YES or NO.
PRESS: ENTER to continue RETURN to exit HELP for more information

```

Figure 2-3 Distributed data facility panel

Data sharing and parallel sysplex

The *data sharing* function of DB2 for z/OS enables applications that run on more than one DB2 for z/OS subsystem to read from and write to the same set of data concurrently.

DB2 subsystems that share data must belong to a DB2 data sharing group, which runs on a an IBM Parallel Sysplex® cluster. A data sharing group is a collection of one or more DB2 subsystems that access shared DB2 data.

A *Parallel Sysplex* is a cluster of z/OS systems that communicate and cooperate with each other. The Parallel Sysplex is a highly sophisticated cluster architecture. It consists of two key pieces of technology:

- Coupling facility

Provides specialized hardware, specialized high-speed links and adapters, and a shared, nonvolatile electronic storage for fast intersystem data sharing protocols.

- Sysplex Timer

Provides a common time source across all the systems in the cluster, thereby delivering an efficient way to provide log-record sequencing and event ordering across the different systems.

The coupling facility and the Sysplex Timer are exclusive to the z Systems environment. They provide strong performance and scalability in a multi-system clustered DBMS environment with shared disks.

Because the DB2 catalog is shared by all members of a data sharing group, data definition, authorization and control are the same as for non-data sharing environments. Be sure that every object has a unique name, and be sure that the shared data resides on shared disks.

Use the same authorization mechanisms that are in place for non-data sharing DB2 subsystems to control access to shared DB2 data and to members. Because all members in the group share the same DB2 catalog, an authorization ID has the same granted privileges and authorities for every member of the group.

Each DB2 subsystem that belongs to a particular data sharing group is a member of that group. All members of a data sharing group use the same shared DB2 catalog.

Figure 2-4 shows that all members of a data sharing group can participate in processing a single query, `SELECT * FROM ACCOUNT`.

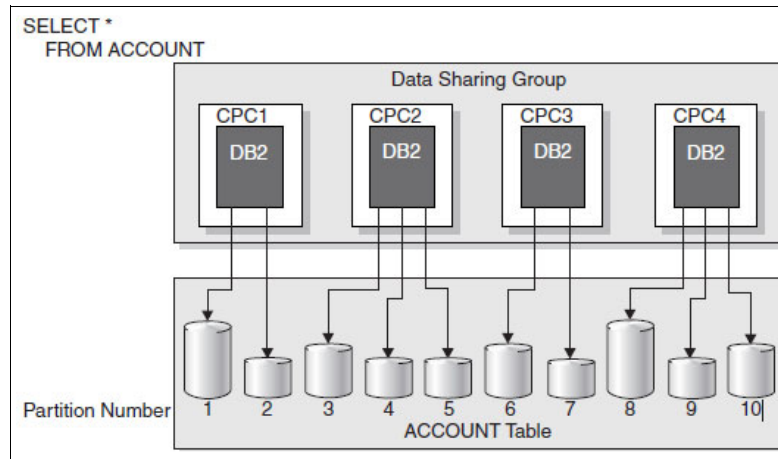


Figure 2-4 Query processed in parallel by member of a data sharing group

Because data sharing provides multiple paths to data, a member can be down, and applications can still access the data through other members of the data sharing group.

As illustrated in Figure 2-5, when an outage occurs and one member is down, transaction managers are informed that the member is unavailable, and they can direct new application requests to another member of the group.

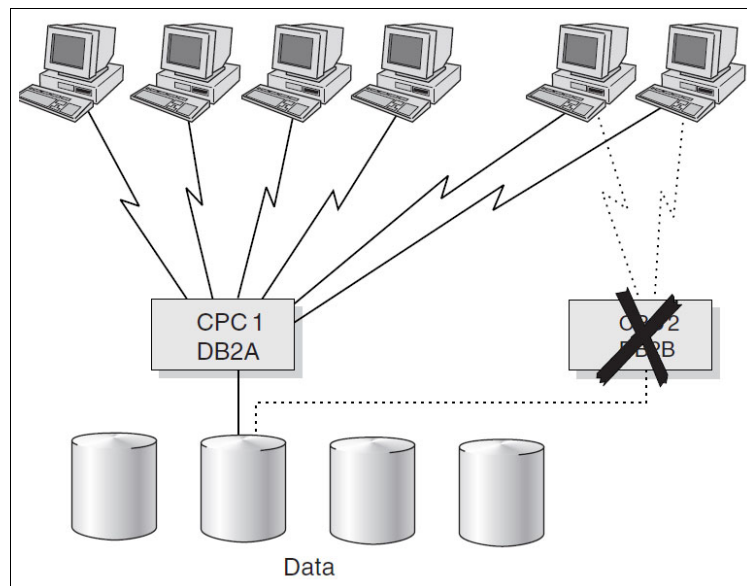


Figure 2-5 Data availability during outages

Before implementing DB2 data sharing, ensure that your security management system supports security in a parallel sysplex environment.

Event monitoring and recording

This section explains how SMF, DB2 internal logging and external security such as RACF can help provide a more complete picture about every process and transaction in DB2:

► SMF

The IBM System Management Facility (SMF) is a component of IBM z/OS operating system. It provides a standardized method of writing records of activity to a file (data set). SMF provides full instrumentation of all baseline activities running on z/OS, including I/O, network activity, software use, error conditions, and processor use.

DB2 generates three types of records in the SMF log:

- Type 100 (DB2 statistics)
- Type 101 (DB2 accounting)
- Type 102 (DB2 performance)

The z/OS DSN1SMFP utility also processes DB2 trace data into reports that are useful for evaluating and auditing. DSN1SMFP accepts data that SMF collects in standard SMF format and produces from one to eighteen reports. DSN1SMFP accepts all SMF record types, but it processes only type 101 (DB2 Accounting) and 102 (DB2 Performance) records. DSN1SMFP checks each type 101 and 102 record for the following DB2 audit trace types:

- 140: Audit Authorization Failures
- 141: Audit DDL Grant/Revoke
- 142: Audit DDL Create/Alter/Drop
- 143: Audit First Write
- 144: Audit First Read
- 145: Audit DML Statement

Because SMF processes volumes of records from DB2, the log can become very large. There can be significant space savings by compressing the DB2 SMF records. **DSNZPARM SMFCOMP**, new since DB2 V10, activates compression of DB2 trace records destined for SMF. If using IBM zEnterprise® Data Compression (zEDC) for z/OS, the IBM Encryption Facility for z/OS can be used to encrypt the compressed data.

► DB2 internal logging

DB2 audit policies are created and stored in the **SYSIBM.SYSAUDITPOLICIES** table. Each policy is specified with specific audit categories and creates a DB2 Instrumentation Facility Component identifier (IFCID) record. For more information about these records, see “Audit category” in the IBM Knowledge Center:

<http://ibm.co/1j433FA>

► External security managers

RACF is referred to here as the external security manager. Except for explicit RACF examples, the general discussion applies equally to any functionally equivalent non-IBM external security manager.

The RACF access control module allows you to use RACF in addition to DB2 authorization checking for DB2 objects, authorities, commands, and utilities.

You can use the log information to link DB2 IFCID records and corresponding RACF SMF records.

As an example, denied access attempts due to inadequate DB2 authorization creates an IFCID140 record, RACF authentication failures create IFCID 83 records and SMF101 is issued indicating the start and stop times.

For more information, see *DB2 10 for z/OS (RACF Access Control Module Guide)*, SC19-2982 and the IBM Redbooks publication titled *Security Functions of IBM DB2 10 for z/OS*, SG24-7959.

2.1.2 Guidelines for configuring security

Data server security threats can be divided into four broad categories: data threats, configuration threats, and audit threats.

- Data threats

Threats against data are mechanisms whereby data can be accessed by users or processes that are not authorized to access such data. This is by far the largest category of threats, and is usually the first that comes to mind. These threats can be aimed directly at the tables in the database, or through more indirect means, such as by looking at the log files or directly at the table space files on the operating system.

- Configuration threats

Threats against configuration are mechanisms whereby the database or database manager configuration files can be tampered with. Because they control critical aspects of your data server — such as how and where authentication is performed — it is critical that the database configuration files are protected as securely as the data itself.

- Audit Threats

Threats against the audit facility are mechanisms whereby the audit configuration, audit logs, or archive logs can be tampered with. In many cases, audit records are the only way to determine *what has happened* in the past and the only form of evidence to detect misuse; it is critical that they be able to withstand tampering.

Some common DB2 security considerations that can help to mitigate these threats include using authentication and authorization methods that adhere to the principle of least privilege. The principle of least privilege means giving a user (or user ID) only those privileges that are essential to the work that is needed. For example, a backup user does not need to install software: hence, the backup user has rights only to run backup and backup-related applications. Any other privileges, such as installing new software, are blocked. To apply this principle to DB2:

- Set proper privileges and access control (such as multilevel security (MLS)) on sensitive data.
- Audit user access, particularly to sensitive data and actions by privileged users, such as a database administrator.
- Revoke the data access authority from the database administrator if they have no business need to access data.
- Limit access given to PUBLIC.
- Remember to protect the staging tables.
- Using trusted contexts in multitier environments.
- Encrypt data *and* backup files at the operating system level.
- Use SSL/TLS to transmit data securely in the network.
- Use operating-system controls to prevent operating-system administrators from gaining too much access.
- Deploy an external data activity monitoring solution if database owners and administrators are reluctant to enable full DB2 logging. You need a full audit trail on all data access operations when you want to deploy a centralized security intelligence solution.

2.2 IBM Information Management System

The Information Management System (IMS) is composed of two parts: IMS Database Manager and IMS Transaction Manager. The IMS Database Manager manages the physical storage of records in the database, and IMS Transaction Manager manages the terminal network, the input and output of messages, and online system resources.

This section describes how the security aspects are managed for IMS databases.

2.2.1 Security concepts and architecture

Database security involves data access or user verification which determines how to establish that the person using an online database is in fact the person that has been authorized. It also involves processing authority or user authority which controls what is seen and what can be done with what is seen after a user's identity is established and verified.

IMS resources can be protected by one or more security facilities such as an external security management program (for example, RACF) or user exit routines.

In this section we discuss IMS security from the perspective of the following categories:

- ▶ Authentication
- ▶ Authorization
- ▶ Data integrity and confidentiality
- ▶ Audit

Authentication and authorization

IMS is able to determine whether a client or a person is who or what it declares to be using an external security manager such as RACF.

IMS talks to RACF through the z/OS SAF interface. The security product does not have to be RACF, other external security managers can invoke SAF, such as Top Secret. CA ACF2 intercepts all calls to the IBM System Authorization Facility (SAF) and processes these requests for security services.

- ▶ SAF is part of the base z/OS Operating System.
- ▶ SAF is invoked at many points within the z/OS Operating System where security decisions must be made.
- ▶ SAF is invoked to perform Authentication, Resource Access control, and Auditing.

Authorization checking is done for IMS resources, such as databases, segments, fields, transaction or other resources, that are requested during source-transaction processing, when the application issues a **CHNG** call and an **AUTH** call and performs a deferred conversational program-to-program message switch. An Authorization (**AUTH**) call verifies each user's security authorization. It determines whether a user is authorized to access the resources specified on the **AUTH** call.

The **SECURITY** macro was implemented in IMS mainly for the purpose of specifying the installations security choices to an external security product, such as RACF.

IMS Version 12 is the last version to support the **SECURITY** macro. You can use initialization parameters to specify all of the **SECURITY** macro keyword values. To reduce the dependency on system generation, IMS Version 13 removes the support for the **SECURITY** stage-1 system definition macro.

Most security options that were previously specified by using the SECURITY macro are now specified by using the following initialization parameters ISIS, RCLASS, RCF, SECCNT, SGN, and TRN.

With the removal of the SECURITY macro in IMS Version 13, you no longer need to specify the use of the Signon/off Security exit routine (DFSCSGN0) and the Transaction Authorization exit routine (DFSCTRN0) during system definition or system startup. Instead, if the exit routines are linked in one of the STEPLIB or LINKLIST libraries, IMS loads the exit routine. There are no startup parameters to specify to load the routines. Message DFS1937I is issued when an exit routine is loaded.

For more information about IMS security enhancements, see “Removal of the IMS Security macro” in IBM Knowledge Center:

<http://ibm.co/1hf28jK>

Before an application program can use the database, you must tell IMS the application program's characteristics and use of data and terminals. You tell IMS the application program characteristics by coding and generating a program specification block (PSB).

The PSB describes the way a database can be viewed by an application. It specifies the database segments an application program can access and also it specifies the functions it can perform on the data such as read, update or delete. PSBs are composed of one or more program communication blocks (PCBs)- one for each database that is to be accessed.

To restrict the scope of data access, use PCBs. A PCB defines a program's (and therefore the user's) view of the database. The PCB can be thought of as a “mask” over the data structure defined by the database definition (DBD), hiding certain parts of it. Therefore, it is possible, simply by limiting the scope of the PCB, to limit the user's access to (and even knowledge of) elements of the database you need to restrict.

For more information about PSBs and PCBs, see “Coding program specification blocks as input to the PSBGEN utility” in the IBM Knowledge Center:

<http://ibm.co/1Vt4f0C>

After you have controlled the scope of data that a user has access to, you can also control authority within that scope. Controlling authority allows you to decide what processing actions against the data a given user is permitted. You can do this through the PROCOPT parameter of the SENSEG statement and through the PCB statement. The PROCOPT statement tells IMS what actions you permit against the database. A program can do what is declared in the PROCOPT.

One potential security exposure is from people attempting to access IMS data sets with non-IMS programs. There are two methods of protecting against this exposure:

- ▶ Data set password protection
- ▶ Database encryption

Password protection

To protect your data, you can take advantage of VSAM password protection to prevent non-IMS programs from reading VSAM data sets on which you have your IMS databases.

To specify password protection for your VSAM data sets, code PASSWD=YES on the DBD statement. IMS then passes the DBD name as the password. If you specify PASSWD=NO on the DBD statement, the console operator is prompted to provide a password to VSAM each time the data set is opened.

Note: This method is only useful in the batch environment, and VSAM password checking is bypassed entirely in the online system. (If you have an external security manager such as RACF installed, you can use it to protect VSAM data sets.)

Encryption

Encrypting the databases is another precaution that can be taken against non-IMS programs reading data language interface (DL/I) databases. IBM IMS offers data encryption capability through IMS segment edit/compression exit routines, for example. By including the IBM Programmed Cryptographic Facility within your exit routine, you can reduce your programming effort. The facility is executed by assembler macro calls. Segments are encrypted before being placed in the database buffer pool. The SEGM control statement in the IMS DBDGEN includes a keyword to specify the name of this exit routine.

You can write a segment edit/compression exit routine to encode and decode segments for security purposes. The logic for data encoding and decoding can be based on information contained within the user-written routine itself. It also can be based on information from an external source, such as data provided in the DBD block, or from tables examined at execution time. The segment edit/compression exit routine is optional.

For more information about encrypting IMS data, see “Encryption: an alternative to access control” in the IBM Knowledge Center:

<http://ibm.co/1KQd6Hi>

Data integrity

Data integrity involves the assurance that the data being accessed or read has not been tampered with, altered, nor damaged since the last authorized access. Integrity involves maintaining and assuring the accuracy and consistency of data throughout its lifecycle.

Both IMS and DB2 for z/OS make it possible for more than one application program to access the data concurrently without endangering the integrity of the data.

To access data concurrently while protecting data integrity, IMS and DB2 for z/OS prevent other application programs from accessing segments that your program deletes, replaces, or inserts, until your program reaches a commit point. A commit point is the place in the program's processing at which it completes a unit of work. When a unit of work is completed, IMS and DB2 for z/OS commit the changes that your program made to the database. Those changes are now permanent and the changed data is now available to other application programs.

To learn more, see “How IMS protects data integrity: commit points” in the IBM Knowledge Center:

<http://ibm.co/1WADM3V>

Integrity and consistency checker

Additionally, IBM offers a suite of IMS Tools, including the IBM IMS Library Integrity Utilities for z/OS. The IMS Library Integrity Utilities provides utilities to validate, compare, map, report, and recover IMS libraries. The product helps you in managing data for the libraries, such as DBD libraries, PSB libraries, ACB libraries, and RECON data sets that you use when referring to IMS databases.

The Integrity Checker protects IMS databases from corruption caused by using the wrong IMS control blocks for access. For instance, if the IMS control blocks used to access the database are not the same as the ones that were used to load the database, then a data integrity exposure can occur. Integrity Checker minimizes this exposure by preventing both batch programs and IMS subsystems from using the incorrect IMS control blocks.

The Consistency Checker ensures that the necessary definition in an IMS subsystem has been created for your database. For a database description (DBD) in the DBD library, Consistency Checker verifies whether the definitions are consistent with the DBD and created appropriately in each library. Consistency Checker generates reports after verification and helps you identify which definitions are needed before you start an IMS subsystem.

Auditing

When a user attempts to access the system, and there is an external security manager installed such as RACF, IMS calls it to check authorization. In section 2.2.1, “Security concepts and architecture” on page 41, recall that SAF is invoked to perform authentication, resource access control and auditing. After RACF has completed its actions, it passes return code and reason code information back to SAF. SAF returns control to the Resource Manager, with its own return and reason codes, and with RACF’s return and reason codes. Each resource access violation creates a RACF type 80 record.

Depending on details about the user, system settings, and resource, logging might take place. RACF calls SMF to perform the logging. SMF is responsible for placing the log records onto the SMF log data sets. You can use the RACF report writer to create reports based on these records.

IMS records the following security violation attempts in the IMS system log:

- ▶ Input message from an unauthorized terminal
- ▶ Password omitted when one is required
- ▶ Password incorrect for authorization
- ▶ Misspelled password
- ▶ Rejected signon
- ▶ Unauthorized **DL/I** command (CMD) call from application program

IMS rejects invalid input messages by sending a message to the terminal entering the message and logging the violation. The IMS system log provides an audit trail for investigation of possible security problems. The IMS system log security violation is identified as an X'10' log record type. You can use the File Select and Formatting Print utility to print the log.

You can reduce the number of notifications caused by operator errors (such as password omitted or misspelled passwords), while still providing evidence of real attempts to avoid security safeguards, by specifying a notification threshold. When the number of violations from a single terminal equals the notification threshold value (as specified by the **SECCNT** initialization EXEC parameter), the master terminal is notified.

External audit management tools

There are other external auditing management applications available such as The IBM IMS Audit Management Expert for z/OS.

The IBM IMS Audit Management Expert for z/OS tool collects and correlates data access information from IMS Online regions, IMS batch jobs, IMS archived log data sets, and SMF records to produce a comprehensive view of business activity for auditors.

IMS Audit Management Expert tool is helpful in finding out how and when data was modified.

IMS Audit Management Expert provides the following features and functions:

► Data collection

IMS Audit Management Expert can collect and correlate many different types of information into its audit repository:

- Access to database data sets and image copy data sets as recorded in SMF
- Access to databases as recorded in the IMS log
- User access to the IMS system through SIGNON as recorded in the IMS log
- PSB and database change of state activity as recorded in the IMS log
- System STOP and START activity as recorded in the IMS log

► Reporting user interface

Provides auditors with flexible options for examining the data in the audit repository

► Administration user interface

Provides administrators with flexible options for user management, auditing profiles, and reporting authorizations

Another tool that can be used for auditing is InfoSphere Guardium S-TAP for IMS which can gather audited events from the following sources:

- IMS database DLI calls performed from within IMS Online Control regions and DLI/DBB batch jobs.
- InfoSphere Guardium S-TAP for IMS can filter audit events generated by database DLI calls by the following call types: Read, Update, Insert, and Delete.
- SMF records.
- InfoSphere Guardium S-TAP for IMS allows the filtering of audit events generated by access methods outside of IMS DLI services, including z/OS access methods such as VSAM or QSAM requests generated from z/OS batch jobs or TSO.
- IMS Log records from IMS System Log data sets (SLDS).
- InfoSphere Guardium S-TAP for IMS allows the filtering of audit events generated by IMS Online Control regions which are logged to IMS log data sets and are processed from within the AUILSTC started task.

Security setup and preparation

In this section we provide an overview on how to set up the various IMS securities facilities that can be used to prevent unauthorized access to databases.

Setting Up RACF

The System Authorization Facility (SAF) in z/OS supports RACF. To set up RACF for security, perform the following actions:

1. Define Resource Classes in the Class Descriptor Table (CDT).
2. Activate Resource Classes.
3. Populate the RACF database:
 - a. Add group and user profiles.
 - b. Connect users to groups.
 - c. Define or update resource profiles and Create access lists.
 - d. Give the correct access authority (NONE, EXECUTE, READ, UPDATE, CONTROL, and ALTER) to a user or group.
 - e. READ is sufficient for most IMS general resources.

Application-based security

Any z/OS application program running in a z/OS address space that is managed by the z/OS Resource Recovery Services (RRS) can access IMS full-function databases and data entry databases (DEDBs). z/OS application programs that use the open database access (ODBA) interface are called ODBA applications.

You need to specify the following parameters to activate RACF:

- ▶ **ODBASE=Y** activates RACF for Allocate PSB calls and
- ▶ **ISIS=R** activates RACF for dependent region access to IMS online

If you specify the parameter **ODBASE=Y** on the IMS execution proclib member, DFSPBxxx, and use RRS, the Open Database Manager (ODBM) checks the user's authority to use the PSB by checking profiles in the defined class. This step is called APSB security, and was available for ODBA in earlier IMS versions.

Encryption

Encrypt data in IMS by creating an encryption exit routine, which is an IMS Segment Edit/Compression exit routine. You can implement encryption in one of two ways:

- ▶ Without compression if you create only an encryption exit routine
- ▶ With compression if you link the encryption exit routine that you create to an existing compression exit routine

For more information, see “IMS encryption and decryption” in the IBM Knowledge Center:

<http://ibm.co/1N6nmP2>

IMS default security

If you do not specify any security in any of the three system definition macros, IMS provides a basic level of resource security called *default security*, which has the following characteristics:

- ▶ It prohibits the entry of certain commands from any terminal other than the master terminal. This basic security function is activated upon completion of Stage 2 of IMS system definition. When you implement input-access security with RACF, IMS removes the default security restrictions. In the case of static terminals, if sign-on is not required, IMS uses the control region user ID for command validation.
- ▶ It applies only to statically defined terminals. Terminals that are defined by using ETO are automatically governed by an identical level of default security. When you modify and use the Command Authorization exit routine (DFSCCMD0), IMS removes the default security for dynamically defined terminals.

For more information, see the following web pages in IBM Knowledge Center:

- ▶ Defining security during DB/DC and DCCTL system definition

<http://ibm.co/1VBW0Kd>

- ▶ Defining security during DB/DC and DCCTL system definition

<http://ibm.co/1VBW0Kd>

Physical security

It is important to consider physical security measures that support your system security, including the following measures:

- ▶ Controlled access to and from the computer area
- ▶ Authorization of data processing operations and non-operations personnel in certain terminal areas
- ▶ Separately controlled areas for media such as tapes, disks, cards, or files
- ▶ Control of computer forms and printed output

Physical security needs are likely to change and require periodic reviews and adjustments.

IMS exits

In this section, we introduce some exit routines used to help secure IMS databases:

- ▶ Data Capture exit routines

Data Capture exit routines capture segment-level data from a DL/I database for propagation to DB2 for z/OS databases. Installations running IMS and DB2 for z/OS databases can use Data Capture exit routines to exchange data across the two database types.

The Data Capture exit routine is an extension of the application program with the same capabilities as the application program. Therefore, the exit routine and the application have equal authorization and limitations. IMS and DB2 resources that the exit routine uses must be authorized in the application program's IMS PSB or DB2 plan. This assures that the application program can access any IMS or DB2 data that is available to the exit routine¹.

The Data Capture exit routine is an installation-written exit routine that can be written in assembler language, C, COBOL or PL/I. The following database types support Data Capture exit routines:

- HISAM
- SHISAM
- HDAM
- PHDAM
- HIDAM
- PHIDAM
- DEDB

The data and the exit routine operate in unprotected, key-8 storage. The exit routine is able to modify data or control blocks that can affect the successful operation of the application program. The data passed to the exit routine is the physical segment data.

- ▶ Segment Edit/Compression exit routines

You can write a Segment Edit/Compression exit routine to compress and expand segments of data. Segment compression saves space and can result in reduced logging. You can write an exit routine for these purposes:

- To edit or compress both fixed- and variable-length segments
- To accomplish either data edit/compression (DEDBs or full-function databases) or key edit/compression (full-function databases only)

¹ http://www.ibm.com/support/knowledgecenter/SSEPH2_13.1.0/com.ibm.ims13.doc.err/ims_db2cdcex.htm

If you write your own exit routine, you can also allow for editing, such as encoding and decoding segments for security purposes, and for validating and formatting data. The logic for data encoding and decoding (or for other editing or formatting) can be based on information contained within the user-written routine. It also can be based on information from an external source, such as data provided in the DBD block, or from tables examined at execution time.

Segment compression is possible for both full-function databases and data entry databases (DEDBs). You can use either DFSCMPX0 or DFSKMPX0, write your own, or generate one which invokes hardware data compression.

Using a dictionary to help establish security

A dictionary, such as the IBM DB/DC Data Dictionary, monitors relationships among entities. This makes it an ideal tool to administer security.

You can use the dictionary to define your authorization matrixes. By using the extensibility feature, you can define terminals, programs, users, data, and their relationships to each other. In this way, you can produce reports that show dangerous trends, who uses what from which terminal, and which user gets what data. For each user, the dictionary can be used to list the following information:

- ▶ Programs that can be used
- ▶ Types of transactions that can be entered
- ▶ Data sets that can be read
- ▶ Data sets that can be modified
- ▶ Categories of data within a data set that can be read
- ▶ Categories of data that can be modified

Adapters and interfaces

The Database Resource Adapter (DRA) is an interface to IMS DB full-function databases and data entry databases (DEDBs). Your application designer can create a program so that the DRA can be used by a coordinator controller (CCTL) or a z/OS application program that uses the Open Database Access (ODBA) interface.

Use the database resource adapter (DRA) startup table to define parameters for the coordinator control (CCTL) region. The DRA startup table establishes the subsystem name of the IMS that is being connected to and defines the characteristics of the DRA, such as the number of threads.

Figure 2-6 is an example of accessing IMS databases from a DB2 for z/OS stored procedure by using the classic Java APIs for IMS, ODBA, and DRA.

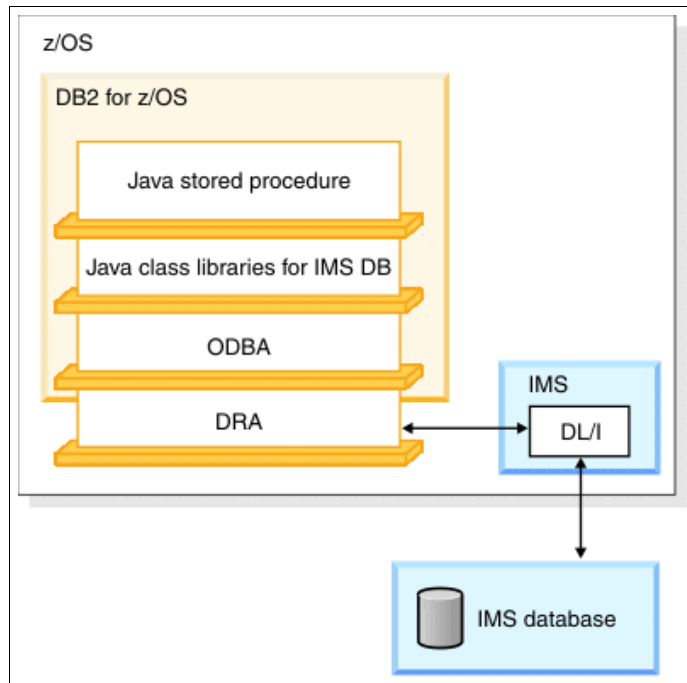


Figure 2-6 Access IMS databases from DB2 using DRA adapter

IMSplex

An IMSplex is made up of IMS and z/OS components that work together. Therefore, the following definitions apply:

- ▶ A set of IMS systems working together to share resources or message queues (or both) and workload
- ▶ A single IMS system using the Common Service Layer (CSL) without the Resource Manager (RM) to have a single point of control

This configuration allows you to use IMS Type 2 commands, which can be issued only through the Operations Manager (OM) APIs by an automated operator program (AOP). Optionally, IMSplexes of this type can also include the CSL Open Database Manager (ODBM).

- ▶ A single IMS system using the CSL with an RM

Comparing the description of an IMSplex with the description of an IBM Parallel Sysplex, we find that a *sysplex* is multiple z/OS images working together, connected by a coupling facility. One or more IMSplex systems can be defined on one or more z/OS images; however, you do not have to have an IMS instance on every z/OS image in the sysplex.

The concept of *data sharing* is important in an IMSplex, specifically in an IMSplex that shares resources and workload. The types of security issues that exist for a stand-alone IMS system also exist for an IMS system within an IMSplex. You must make the following decisions when implementing security for IMS:

- ▶ Determine what resources need to be protected.
- ▶ Determine which users need access to the resources.
- ▶ Determine the level of access to a resource that the users need.

In addition to the security checks made for stand-alone IMS systems, an IMSplex with a common service layer (CSL) performs other security checks. The CSL includes the following components:

- ▶ Open Database Manager (ODBM), which provides access to IMS databases managed by the IMS DB systems in DBCTL and DB/TM environments within the IMSPlex.
- ▶ Operations Manager (OM), which routes commands, consolidates command responses, provides an API for command automation, and provides user exits for customization and security.
- ▶ Resource Manager (RM), which maintains global resource information, ensures resource consistency, and coordinates IMSplex-wide processes such as global online change.
- ▶ Structured Call Interface (SCI), which routes messages, registers and unregisters members of an IMSplex, and provides security authentication of IMSplex members.

To access the IMSplex, clients must first register with the SCI, which allows IMSplex members to communicate with one another. Communication between IMSplex members can occur within a single z/OS image or among multiple z/OS images. The individual IMSplex members do not need to know where the other members reside or what communication interface to use.

A client can register with SCI only if the user ID of the address space in which the client is running has the authority to do so. SCI checks security when a client issues a request (CSLSCREG) to register with SCI. The SCI address space registers with itself but does not need security authorization.

Figure 2-7 shows an example of single point of control (SPOC) automation that uses an IMSplex of a single IMS control region communicating through the SCI to various other functions.

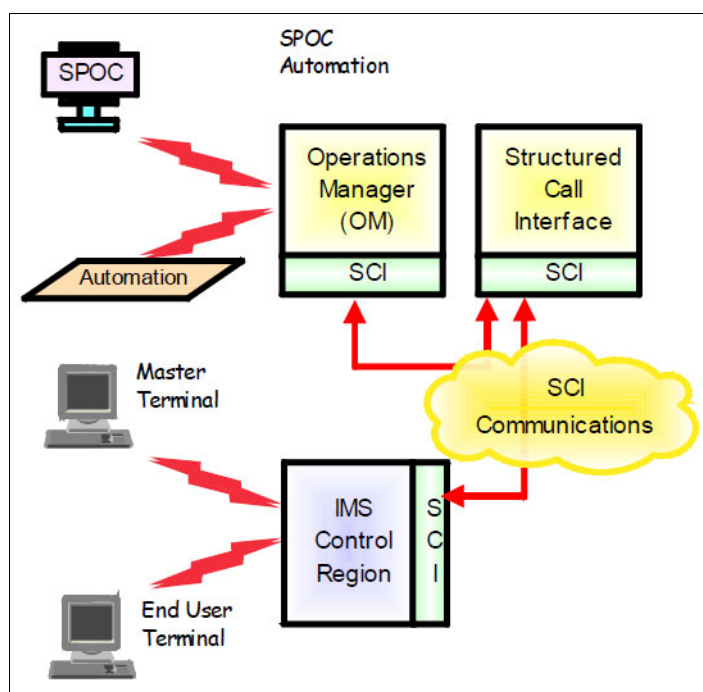


Figure 2-7 IMSplex

A sample request authorization exit called IPIUAUTH is also supplied. The exit is called by DBRC Log Selection to validate the IMSplex specification. The exit as supplied (IPIUAUTH in the SIPISAMP library) stops all DBRC requests if IMSplex is specified. The exit is optional. If it is not installed, all requests are allowed. The sample library SIPISAMP contains the sample program (IPIUAUTH) and a JCL member (IPIU002) for a Request Authorization exit.

2.2.2 Guidelines for configuring security

Before you decide which security facilities to use in designing a secure IMS system, make sure that you understand what to protect before you decide how to protect it. When considering each part of your security design, consider the physical actions that a user must take to get access to the system. Typically, more than one type of security check should be involved. An external security manager, such as RACF, should also be involved.

These are the primary security threats to IMS:

Terminal access

Users must be forced to identify themselves each time they use a terminal. Users should be authorized for only the set of transactions and commands that they need. Signon verification security requires the entry of a user ID as a parameter on the **/SIGN** command at all terminals or a subset of terminals.

If you are using RACF, a RACF PassTicket is a one-time-only password that is generated by a requesting product or function. It is an alternative to the RACF password and removes the need to send RACF passwords across the network in clear text.

RACF offers a terminal-user security function that ranges from no security for a particular terminal to permitting a certain predefined list of users access through a physical control point. A terminal-user profile can be created for every PTERM in the IMS.

Master terminal access

Because the master terminal can modify all security profiles during normal operations, it is imperative to secure this terminal with a second level of control. Signon verification security provides this capability. The primary question is how much capability to modify security should be given to this second level of control.

Default security does not and cannot prevent modifying the system's security profiles through the master terminal. However, you might want to restrict some commands from being entered from the master terminal. Use the DFSCCMD0 exit routine to limit the commands that can be entered.

Automated operator programs

Automated operator (AO) application programs can issue a subset of IMS operator commands. Because an operator command can compromise a security profile, you should prevent AO application programs from issuing sensitive commands that might modify IMS resources.

Regions It is important to secure DB/DC- and DCCTL-dependent regions and their resources. There are two approaches to securing a dependent region: Preventing the unauthorized scheduling of application programs in the dependent region and preventing the unauthorized use of resources by those programs after they have been scheduled.

These additional resources can be protected in the DB Control (DBCTL) environment:

Control region	The IMS system control region that enables online application programs to process the database through terminals.
System data set	A collection of data that is fundamental to the operation of the IMS online system. An example is the IMS.PROCLIB data set.
Dependent region	An area of storage in the IMS online system in which batch or online application programs are executed. In a DBCTL environment, the dependent region can be a Batch message programs (BMP) region or a Java message processing (JMP) region.
Program specification block (PSB)	The control block that describes a group of hierarchic databases and logical message destinations used by an online application program.
BMP application program	A program in a DBCTL environment that performs work for a user. Batch message programs (BMPs) are activated by the dependent region controller after the region is started by JCL.
Database	A collection of data that is fundamental to the user's activity. Using a Program Communications Block (PCB), a program has a logical view of the database, as described by the IMS physical database design.

2.3 Virtual Storage Access Method

Virtual Storage Access Method (VSAM) is one of the access methods in z/OS used to process data and applies to both a data set type and the access method used to manage various data set types. It applies only to data stored on direct access storage devices (DASDs).

VSAM is used to organize records into four types of data sets. The primary difference among these types of data sets is the way their records are stored and accessed:

- ▶ **key-sequenced (KSDS)**

In a KSDS organization, records are initially loaded in the data component in ascending collating sequence by key. With VSAM, it is possible to assign as many as 12 different passwords to each VSAM KSDS (four each for the cluster, the data component, and the index component).

- ▶ **entry-sequenced (ESDS)**

In an ESDS organization, records are sequenced by the order of their entry in the data set, rather than by key field in the logical record.

- ▶ linear (LDS)

An LDS contains data that can be accessed as byte-addressable strings in virtual storage. The most common LDS user is IBM DB2.

- ▶ relative record (RRDS)

An RRDS consists of several preformatted, fixed-length logical records slots. Each slot has a unique relative record number (RRN), and the slots are sequenced by ascending relative record number. Relative record data sets are typically directly accessible and therefore, the most vulnerable to security breaches.

For a more thorough comparison of VSAM data set organizations, see the IBM Redbooks publication titled *VSAM Demystified*, SG24-6105.

The SHAREOPTIONS parameter value indicates to VSAM what degree of shared access is granted.

2.3.1 Security concepts and architecture

There are two major parts to VSAM:

- ▶ Catalog management

The structure of VSAM consists of four basic concepts:

- Master catalog
- User catalog
- Data space
- File, which is commonly referred to as a *cluster*

The catalogs maintain the unit and volume information where a data set resides. The catalogs are used for retrieval of data sets. These catalogs are in the form of a KSDS cluster.

The master catalog is a required file which must be defined by using the IDCAMS program. Every file that will be used must be under the control of the master catalog. The information stored for each file allows, among other things, for VSAM to access the file, password authorization, and statistics for the many operations that may be performed on the file.

Upon request from an application program, VSAM locates, opens, and closes a requested file. In doing so, it checks passwords and ensures that the master catalog information for the requested file is correct.

- ▶ Record management

The record management part of VSAM contains the access method code. VSAM is used to organize records into the four main types of data sets, as described in the opening paragraphs in 2.3, “Virtual Storage Access Method” on page 52.

Because of the possibility of multiple passwords used for VSAM security, most installations use other security managers, such as the RACF that provide a more comprehensive, system-wide approach to security.

This section describes the following main security concepts for VSAM:

- ▶ Authentication and authorization
- ▶ Data integrity
- ▶ Auditing

Authentication and authorization

VSAM data sets can be protected using RACF (or other external security manager) by controlling who has authority to access them and at what authority level they can do so. When users attempt to use a data set, they must authenticate with a user ID and password. If authentication is successful, the user profile and the data set profile are checked to decide what kind of access should be granted.

To protect data sets on DASD, define profiles for the data sets that you want to protect. You define profiles to protect two RACF categories of data sets:

- ▶ *Profiles for user data sets*, where the high-level qualifier is a RACF user ID. All RACF-defined users can protect their own data sets.
- ▶ *Profiles for group data sets*, where the high-level qualifier is a RACF group name (see RACF group profiles for information about RACF groups). A RACF-defined user can RACF-protect group data sets provided the user has the necessary authority or attributes. (See the *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683, for details.)

A data set profile defined RACF contains the following attributes:

- ▶ Data set name
- ▶ Owner
- ▶ RACF access list, which is a list of specific users and groups who can access a data set
- ▶ The default level of access authority granted for all users or groups not specified in the access list
- ▶ Access permissions

You can assign the following access permissions to specific users or groups:

ALTER	Allows users to read, update, or delete the data set
CONTROL	Provides the authority to perform control interval access (access to individual VSAM data blocks) and to retrieve, update, insert, or delete records in the specified data set
UPDATE	Allows users to read or update the data set (but does not authorize a user to delete the data set)
READ	Allows users to access the data set for reading or copying only.
NONE	Does not allow users to access the data set

Anyone who has READ, UPDATE, CONTROL, or ALTER authority to a protected data set can create a copy of it. As owner of the copied data set, that user has control of the security characteristics of the copied data set and can downgrade it. For this reason, you should assign a UACC of NONE, and then selectively permit a small number of users to access your data set as their needs become known.

Data integrity

When you define VSAM data sets, you can specify how the data is to be shared within a single system or among multiple systems that can have access to your data and share the same direct-access devices. Before you define the level of sharing for a data set, you must evaluate the consequences of reading incorrect data (a loss of read integrity) and writing incorrect data (a loss of write integrity). Situations can result when one or more of the data set's users do not adhere to guidelines for accessing shared data sets.²

² http://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.idad400/share.htm

To protect the integrity of your VSAM data sets, VSAM uses the following mechanisms:

- ▶ A single central lock structure using the lock-assist mechanism of the MVS coupling facility
VSAM record-level sharing (RLS) support requires the coupling facility to define a master lock structure for cross-system locking. The VSAM lock mechanism controls data sharing. This central lock structure provides sysplex-wide locking at the record level.
- ▶ Internal locks for non-RLS access (when you are using a single VSAM control block structure)
- ▶ SYSVSAM ENQ issued by VSAM according to the data set SHAREOPTIONS for non-RLS access that involves multiple VSAM control block structures
- ▶ The GRS serialization function issued (ENQ macro) by the data set allocation routine, as requested by the initiator according to the data set disposition at DD statement
- ▶ The ENQ serialization function (can also be implemented by the user task application program)

Encryption

VSAM data integrity and security depend on *encryption*, also known as *enciphered data*.

You can use the **VSAM REPRO** command to encipher data that is written to a data set, and then store the enciphered data set offline. When wanted, you can decipher the data set and use the **REPRO** command to decipher the encrypted data. You can decipher the data either on the host processor on which it was enciphered or on another host processor that contains the Access Method Services Cryptographic Option and the same cryptographic key that was used to encipher the data. You can use either ICSF (Integrated Cryptographic Service Facility) or keys that the Access Method Services user supplies to create the cryptographic keys.

Example 2-1 shows a VSAM IDCAMS program that uses the **REPRO** command to encipher the PAOLOR9.CLEAR input data set to produce the PAOLOR9.CRIPT output data set by using the clear key ENCIPHER facility with the manual DATAKEYVALUE.

Example 2-1 Encipher using a clear key in a REPRO command

```
//ENC EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO -
  INDATASET('PAOLOR9.CLEAR') -
  OUTDATASET('PAOLOR9.CRIPT') -
  ENCIPHER(PRIVATEKEY DATAKEYVALUE('211020103'))
```

Example 2-2 shows the VSAM IDCAMS program that is now used with the **REPRO** command and the same clear key to decipher the PAOLOR9.CRIPT data set.

Example 2-2 Decipher using a clear key in REPRO command

```
//ENC EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
INDATASET('PAOLOR9.CRIPT') -
OUTDATASET('PAOLOR9.CLEAR.AGAIN') -
DECIPHER(DATAKEYVALUE('211020103'))
```

Auditing

RACF logs data set use on SMF to provide auditing records for VSAM data sets. SMF record types that are recorded are listed in Table 2-2.

Table 2-2 SMF VSAM record types

Type	Description
60	VSAM Volume data set Updated
62	VSAM Component or Cluster Opened
63	VSAM Catalog Entry Defined
64	VSAM Component or Cluster Status
67	VSAM Catalog Entry Delete
68	VSAM Catalog Entry renamed
69	VSAM Data Space, defined, extended or deleted
80	RACF Processing
81	RACF Initialization
83	RACF Processing Record for Auditing data sets

The auditing granularity can be the use of specific data set or general. For example, you can define the profile to audit only the attempts to update on a specific data set, or, you can set up a resource profile for your data set to audit every attempt to use a data set. There are several ways to look into these SMF records:

- ▶ IBM SystemView Enterprise Performance Data Manager/MVS (EPDM) program
- ▶ DBU2MSXL, a set of scripts that loads the output of the RACF Database Unload Utility (IRRDBU00) into a Microsoft Excel spreadsheet in conjunction with DBU2MSAC, a set of scripts that loads the output of IRRDBU00 into Microsoft Access database
- ▶ DFSORT ICETOOL utility, which can analyze and produce reports from IRRDBU00 output

The DBU2MSXL and DBU2MSAC tools can be downloaded from the following web pages:

- ▶ <http://www.ibm.com/systems/z/os/zos/features/racf/downloads/dbu2msxl.html>
- ▶ <http://www.ibm.com/systems/z/os/zos/features/racf/downloads/dbu2msac.html>

For more information, see *OS/390 Security Server Audit Tool and Report Application*, SG24-4820.

2.3.2 Guidelines for configuring security

In this section, we provide guidelines to some of the security definitions for VSAM record-level sharing (RLS) and Sharing Control data sets (SHCDS). We also examine some of the common VSAM security utilities.

VSAM RLS

After VSAM RLS is set up, you already have defined the necessary lock and cache structures in the coupling facility, so you can now create the following security definitions to implement DFSMSStvs:

- ▶ Define a RACF generic profile to protect all the log streams that are referenced by a DFSMSStvs instance.
- ▶ For all forward-recoverable data sets that DFSMSStvs accesses, grant DFSMSStvs access to the following logs:
 - The log of logs
 - The forward recovery log streams

SHCDS security definitions

In this section, we list actions to perform to optimize the security of your VSAM data sets.

Before an application program can access a data set, it must first issue the OPEN macro to open the data set (after allocation) for processing. OPEN is issued against a user application control block (ACB). Among other things, the OPEN macro checks your program task security authorization for reading and writing through the RACROUTE service (RACF).

To access the SHCDS data set and use the **VARY SHCDS** command, ensure that the following actions are performed:

- ▶ SMSVSAM must be RACF authorized to update SYS1.DFPSHCDS.* data sets. For more information, see “Security definitions” in *VSAM Demystified*, SG24-6105.
- ▶ To use the **SHCDS** command, you must have access to the STGTo facility class to use the access method services **SHCDS** command, and you must be authorized for the facility class STGADMIN.IGWSHCDS.REPAIRADMIN.IGWSHCDS.*.

If you intend to use the **SHCDS** command under TSO, you must add SHCDS to the AUTHCMD NAMES list in your IKJTSOxx parmlib member.

Utilities

In this section, we describe some of the common VSAM security utilities.

Several IDCAMS commands are available to check your VSAM data sets:

▶ EXAMINE

You can use the **EXAMINE IDCAMS** command to analyze and report on the structural integrity of the index and data components of these objects:

- A key-sequenced data set cluster (KSDS).
- A variable-length relative record data set cluster (VRRDS).
- Any problems with the VSAM data set are reported by one of the IDCxxxxx messages.

▶ VERIFY

The **VERIFY** command causes a catalog to correctly reflect the end of a VSAM data set after an error occurs while closing a VSAM data set. The error might cause the catalog to be incorrect.

► **DIAGNOSE**

The **DIAGNOSE** command can be used to scan a basic catalog structure (BCS) or a VSAM volume data set (VVDS) to validate the data structures and detect structure errors.

► **DFSMSdss PRINT**

With the **PRINT** command, you can print the following data:

- A single volume non-VSAM data set, as specified by a fully qualified name. You must specify the volume where the data set is, but you do not have to specify the range of tracks that it occupies.
- A single-volume VSAM data set component (not cluster). The component name that is specified must be the name in the VTOC, not the name in the catalog.
- Ranges of tracks.
- All or part of the VTOC. The VTOC location does not need to be known.

► **LISTCAT**

You can list the catalog entries by using the **LISTCAT** command.



WebSphere Application Servers and web servers

In this chapter, we provide a security concepts and architecture overview for IBM WebSphere Application Server. Also we cover some of the guiding principles for configuring application and web server security and some WebSphere specific configurations developed for IBM z Systems.

This chapter covers the following topics:

- ▶ IBM WebSphere Application Server overview
- ▶ Security concepts and architecture
- ▶ Interfaces (transaction systems, databases, IBM MQ, web server, and other adapters)
- ▶ Guiding principles for configuring security

3.1 IBM WebSphere Application Server overview

As mentioned in Chapter 1, “Introduction to major mainframe middleware components” on page 1, IBM WebSphere Application Server is a Java Platform, Enterprise Edition (Java EE) and web services technology-based application. It was built to support single-server environments and medium-sized configurations to complex web applications that demand excessive processing power and fast response. It supports all major functions, such as servlets, JavaServer Pages (JSPs), and Enterprise JavaBeans (EJBs), including the latest technology integration of services and interfaces.

To get the most you can from this book, you need to be aware of the WebSphere Application Server structure and concepts. Most of these concepts are commonly used by a person in an administrative role, but they can help facilitate any design and troubleshooting situations.

WebSphere Application Server consists of six components:

- Applications

WebSphere Application Server, the main ability to which is to run the following types of applications:

- Java EE applications
- Portlet applications
- Session Initiation Protocol Applications
- Business-level applications
- OSGi applications

- Application servers

The *application server* is where Java language-based applications run. It provides services that can be used by business applications, such as database connectivity, threading, and workload management. At the core of each product in the WebSphere Application Server family, there is an application server. The following IBM WebSphere Application Server products are among those mentioned in this book:

- Express
- Base
- Network Deployment

These all have essentially the same architectural structure with some minor differences. Both Base and Express platforms are limited to stand-alone application servers. The Network Deployment edition provides a more advanced topology that enables you to use features such as workload management, scalability, high availability, and central management of multiple application servers.

- Profiles

A *profile* can define a deployment manager, a stand-alone application server, or an empty node to be federated (added) to a cell. It can be created during and after the installation. WebSphere Application Server runtime environments are built by creating profiles. Each profile contains files that are specific to that run time (such as logs and configuration files).

Profiles are sets of files that represent a WebSphere Application Server configuration. There are two categories of WebSphere Application Server files available:

- *Product files* are a set of read-only static files or product binary files that are shared by any instances of WebSphere Application Server product.
- *Configuration Files (profiles)* are a set of user-customizable data files including WebSphere configuration, installed applications, resource adapters, properties, log files and so on.

Administration is greatly enhanced when using profiles rather than multiple product installations. Also, you save disk space and follow a simplified process when updating the product. Creating new profiles is more efficient and less prone to error than full product installations. This enables developers to create separate profiles of the product for development and testing.

Each profile is stored in a unique directory path, which is selected by the user at profile creation time. Profiles are stored in a subdirectory of the installation directory by default, but they can be located anywhere.

- Nodes, node agents, and node groups

A *node* is an administrative grouping of application servers for configuration and operational management within one operating system instance. You can create multiple nodes within one operating system instance, but a node cannot leave the operating system boundaries. A stand-alone application server configuration has only one node. With Network Deployment, you can configure a distributed server environment that consists of multiple nodes, which are managed from one central administration server.

In distributed server configurations, each node has a *node agent* that works with the deployment manager to manage administration processes. A node agent is created automatically when you add (federate) a stand-alone node to a cell. Node agents are not included in the Base and Express configurations because a deployment manager is not needed in these architectures. The node agent is responsible for monitoring the application servers on the node and routing administrative requests from the deployment manager to those application servers.

A *node group* is a grouping of nodes, within a cell, that have similar capabilities. A node group validates that the node can perform certain functions before allowing them. For example, a cluster cannot contain both z/OS nodes and nodes that are not z/OS-based. In this case, you can define multiple node groups: One for the z/OS nodes and one for nodes other than z/OS. A DefaultNodeGroup is created automatically. It contains the deployment manager and any new nodes with the same platform type. A node can be a member of more than one node group.

- Cells

A *cell* is a grouping of nodes into a single administrative domain. A cell encompasses the entire management domain. In the Base and Express configurations, a cell contains one node, and that node contains one server. In a Network Deployment environment, a cell can consist of multiple nodes (and node groups), which are all administered from a single point, the deployment manager. A cell configuration that contains nodes that are running on the same platform is called a *homogeneous cell*. It is also possible to configure a cell that consists of nodes on mixed platforms.

- Deployment manager

The *deployment manager* is the central administration point of a cell that consists of multiple nodes and node groups in a distributed server configuration. The deployment manager communicates with the node agent to manage the applications servers within one node. A deployment manager provides management capability for multiple federated nodes and can manage nodes that span multiple systems and platforms. A node can be managed only by a single deployment manager, and the node must be federated to the cell of that deployment manager. The configuration and application files for all nodes in the cell are centralized into a master configuration repository. This centralized repository is managed by the deployment manager and synchronized with local copies that are held on each of the nodes.

There are other concepts that we do not cover because they are not relevant to the focus of this book. You can find a detailed description of these concepts and the ones described in this section in the IBM Redbooks publication titled *IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide*, SG24-7957.

WebSphere Application Server for z/OS works a bit differently from the versions on distributed platforms. For example, it is capable of using workload load balancing and response time goals on a transactional base and on a special clustering mechanism, the multiservant region, with a stand-alone application server.

In distributed platforms, WebSphere Application Server is based on a single process model, which means that the entire application server runs in a single process that contains the Java virtual machine (JVM). Assuming that this application is not clustered, if the process crashes, all applications that are deployed on this application server will be unavailable.

On WebSphere Application Server for z/OS, a logical application can consist of multiple JVMs, where each one executes in a different address space. These address spaces are called *servant regions* (SRs), and each contains one JVM. If a servant region abends, another servant can take over the incoming requests in a multiple-servant environment.

The main difference in the z/OS logical partitions in comparison to distributed platforms is that each logical partition in z/OS has cluster capabilities through the use of these multiple servants and mini clusters that can benefit from cluster advantages, such as availability and scalability, without the overhead of a real cluster.

To implement security over a WebSphere Application Server environment we need at least to understand the way it works and its structure. With that in mind, we can now discuss the security concepts, architecture and some security best practices for WebSphere Application Server on z/OS.

3.2 Security concepts and architecture

Security is an important concern and a critical element of every IBM WebSphere Application Server product. You can use a variety of included features to assure the safety of your application and all of the other resources involved. WebSphere Application Server has the security reasonably configured by default, but more complex environments involve unique issues that simply cannot be anticipated.

Note: Keep in mind that there is no such a thing as a *perfectly* secure system, and this book does not intend to imply otherwise.

These are the topics that we cover in the remainder of this chapter:

- ▶ Global security configuration
- ▶ SSL/TLS
- ▶ Java security
- ▶ Interfaces (transaction systems, databases, IBM MQ, web server, and other adapters)

3.2.1 Global security configuration

Before you start to set the security parameters in your environment, it helps to be aware of several concepts and terms used throughout this section:

- 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, which are a major component of systems failure. Therefore, it is highly desirable but difficult to implement. You can find more information about SSO on the Open Group website:

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

- 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.

- Principal

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

- 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”).

- 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.

Figure 3-1 shows how IBM WebSphere Application Server security is layered.

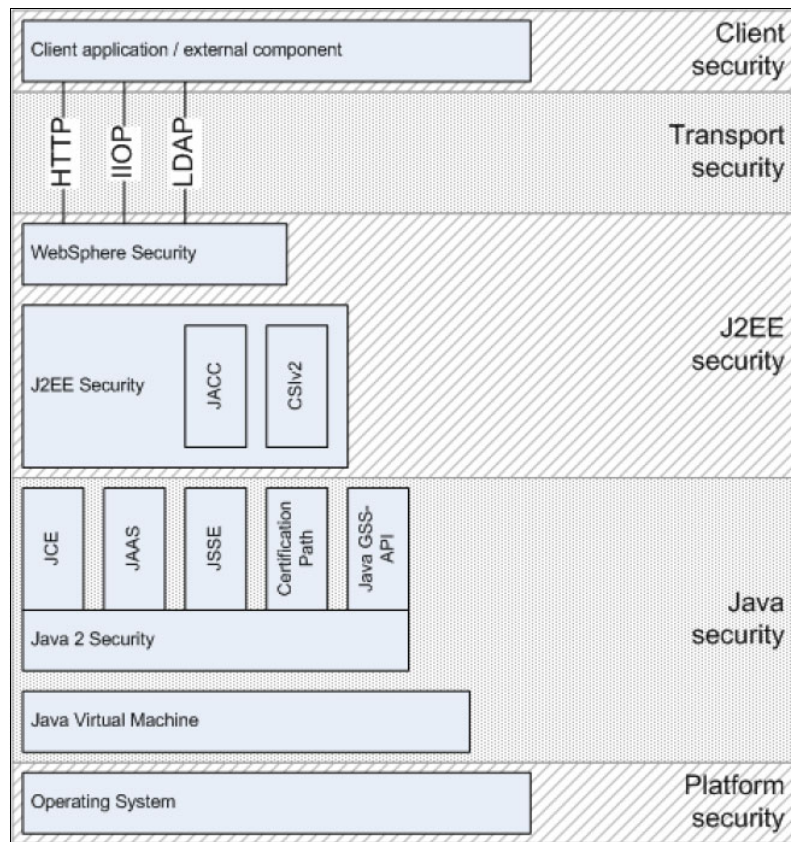


Figure 3-1 WebSphere Application Server security layers

There are many security options to be set and configured in WebSphere Application Server. Some may not apply to your environment due to your application intentions. The security configurations can be set in the console by selecting **Security** → **Global Security**. Figure 3-2 shows the security options in WebSphere Application Server for z/OS.

Administrative security

- ☒ Enable administrative security
 - [Administrative user roles](#)
 - [Administrative group roles](#)
 - [Administrative authentication](#)

Application security

- ☒ Enable application security

Java 2 security

- ☒ Use Java 2 security to restrict application access to local resources
 - ☐ Warn if applications are granted custom permissions
 - ☒ Restrict access to resource authentication data

User account repository

Realm name

Current realm definition

Available realm definitions

Authentication

Authentication mechanisms and expiration

- ☒ LTPA
- ☐ Kerberos and LTPA
 - [Kerberos configuration](#)
 - [Authentication cache settings](#)
- ☐ Web and SIP security
- ☐ RMI/IIOP security
- ☐ Java Authentication and Authorization Service
 - ☐ Enable Java Authentication SPI (JASPI)
 - [Providers](#)
 - ☐ Use realm-qualified user names

- [Security domains](#)
- [External authorization providers](#)
- [Programmatic session cookie configuration](#)
- [Custom properties](#)
- [z/OS security options](#)

Figure 3-2 WebSphere Application Server for z/OS security settings page

To understand what each option does, in general, we provide a brief description of the following topics and what effects they have after they are enabled:

- ▶ Administrative security
- ▶ Application security
- ▶ Java 2 security
- ▶ z/OS security options
- ▶ Security domains
- ▶ User account repository
- ▶ Authentication
- ▶ Java Authentication and Authorization Service (JAAS)

Administrative security

This option is enabled by default as part of the installation process in versions before WebSphere Application Server V6.1. Since then, the concept of *global security* was split into *administrative security* and *application security* (see “Application security” on page 67), and each component can be enabled separately.


WebSphere Application Server provides the ability to secure the administrative consoles so that only authenticated users can use them by enabling administrative security. Administrative security determines whether security is used at all, provides authentication of users by using the WebSphere Application Server administrative function, the type of registry against which authentication takes place, and other values.

Enabling administrative security activates the settings that protect your server from unauthorized users. Enabling administrative security does not enable application security. All profiles except the custom profile can be secured by enabling the administrative security.

Tips for enabling administrative security: An XML file-based user repository is created during profile creation. Later, it can be federated with other repositories to provide a robust user registry for both administrative and application security.

If you do not want to use the file-based repository, do not enable administrative security during profile creation. Configure it manually afterward, instead.

If you enable administrative security during profile creation, you are asked for a user ID and password that are added to a file-based user registry with the Administrator role, as shown in Figure 3-3.



Profile Management Tool 8.0

Administrative Security

Choose whether to enable administrative security. To enable security, supply a user name and password for logging into administrative tools. This administrative user is created in a repository within the application server. After profile creation finishes, you can add more users, groups, or external repositories.

☒ Enable administrative security

User name:
wasadmin

Password:
.....

Confirm password:
.....

See the information center for more information about administrative security.
[View the online information center](#)

< Back Next > Cancel Finish

Figure 3-3 Enable administrative security

Before WebSphere Application Server V6.1, users granted administrative roles and administered all of the resource instances under the cell. Starting with Version 6.1, administrative roles are per resource instance rather than assigned to the entire cell. Resources that require the same privileges are placed in a group called the *authorization group*. Users can be granted access to the authorization group by assigning to them the required administrative role within the group.

A *cell-wide authorization group* exists for compatibility with an earlier version. Users who are assigned to administrative roles in the cell-wide authorization group can still access all of the resources within the cell.

The following administrative security roles are available:

Administrator	The administrator role has operator permissions, configurator permissions, and the permission required to access sensitive data, including server password, Lightweight Third Party Authentication (LTPA) password and keys, and so on.
Auditor	The auditor role has permission to view and change the configuration settings for the security auditing subsystem.
Configurator	The configurator role has monitor permissions and can change the WebSphere Application Server configuration.
Operator	The operator role has monitor permissions and can change the runtime state, for example, the operator can start or stop services.
Monitor	The monitor role has the least permissions. This role primarily confines the user to viewing the WebSphere Application Server configuration and current state.
Deployer	The deployer role has permission to perform both configuration actions and runtime operations on applications.
Admin Security Manager	The Admin Security Manager role gives permissions to users to map other users to administrative roles. When fine-grained administrative security is used, users granted this role can manage authorization groups. A user mapped to the administrator role does not have permissions to map users to administrative roles.
ISC Admin	The ISC Admin role has administrator privileges for managing users and groups from within the administrative console only.

You can find more information about “Administrative roles” in the IBM Knowledge Center:

<http://ibm.co/10YzXV0>

Application security

In releases before WebSphere Application Server V6.1, the application security parameter is disabled by default after the installation process. Application security provides application isolation and requirements for authenticating users for the applications in your environment.

Application security must be enabled if *declarative security* is used by any application that is deployed in the application server. However, if your application relies only on *programmatic security*, for example using the `HttpServletRequest` interface method `getRemoteUser()` where authentication is already done on the Hypertext Transfer Protocol (HTTP) server side, you are not required to enable application security.

Application security implementation requires close interaction and planning among application developers, security specialists, and administrators. Two primary decisions must be addressed during the planning stage for the overall application security design in an enterprise:

- ▶ Programmatic and declarative security
- ▶ Deploying a secured enterprise application

For more information about this topic, see the IBM Redbooks publication titled *WebSphere Application Server V7.0 Security Guide*, SG24-7660.

Java 2 security

For those who are familiar with a System Authorization Facility (SAF) approach to resource protection, the concepts of Java 2 security might seem foreign. This is because Java 2 security was designed to address a problem different from SAF security. SAF security protects resources in an environment shared among multiple users with different levels of privilege. Therefore, resources are defined in profiles, and users (or groups of users) are permitted access to these resources with different levels of authority (such as READ, UPDATE, EXECUTE).

Java 2 security protects resources in an environment where the executable code might not be trustworthy. A classic example is where the user of a personal workstation has downloaded code from the Internet.

Java 2 security was introduced in SDK 1.3. Earlier Java implementations, before Java V1.2, only had the sandbox model, which provided a restrictive environment. With Java V1.2, a new security model was introduced, as Figure 3-4 shows.

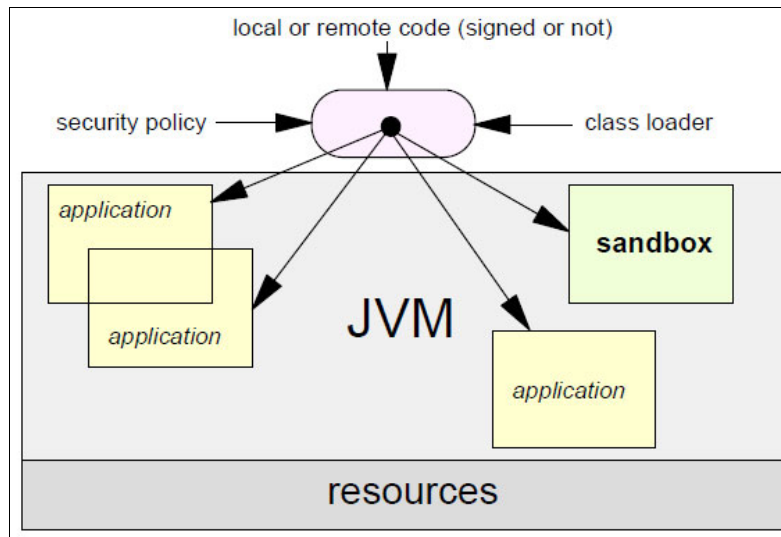


Figure 3-4 Java 2 platform security model

This new model is meant to provide the following security features for the JVM:

- ▶ Fine-grained access control
This was available in the earlier version, which used programmatic access control security.
- ▶ Easy configuration of security policy
It also available in previous versions, and also used for programmatic security.
- ▶ Easy extension for the access control structure
The new architecture allows typed security permissions and provides automatic handling for them.
- ▶ Extension of security checks to all Java programs (both applications and applets)
Every Java code is under security control, which means that local code is no longer trusted by default.

This model is fully supported in WebSphere Application Server. It provides policy-controlled protections of system resources, such as file I/O, sockets connection, program threads, and class loading regarding whether a piece of code can access the resource. The policy is declarative.

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 locked down by RACF. In general, Java 2 security is redundant in a secure environment such as z/OS. Many IBM 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()`, that causes 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.

z/OS security options

The z/OS security option was highlighted in Figure 3-2 on page 65 because this option is only available on WebSphere Application Server for z/OS, and it is important to use it to enforce security. Enabling the z/OS security option on WebSphere Application Server provides the following two security features:

- Enable application server and z/OS thread identity synchronization

Application servers can process the `SyncToOSThread` option for application components that specify it. Selecting this option indicates whether an operating system thread identity is enabled for synchronization with the Java Platform, Enterprise Edition (Java EE) identity that is used in the application server run time when an application is coded to request this function.

- Enable the connection manager RunAs thread identity

This sets the MVS identity associated with the Java EE identity on the execution thread. Local Java EE Connector architecture (J2CA) connectors may honor the MVS identity for authentication and authorization when an application requests a connection. When you enable this setting, the method can process a request that modifies the operating system identity to reflect the Java EE identity. This function is required to take advantage of thread identity support. Java EE Connector architecture (J2CA) connectors that access local resources on a z/OS system can use the thread identity support.

Security domains

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

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

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

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

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

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

Table 3-1 Comparison of global and domain security settings

Global security configuration	Security domain overrides
<ul style="list-style-type: none"> ▶ Enablement of application security ▶ Java 2 security ▶ User realm (registry) ▶ Trust Association Interceptor (TAI) ▶ SPNEGO Web authentication ▶ RMI/IIOP Security (CSlv2 protocol) ▶ JAAS ▶ Authentication mechanism attributes ▶ Authorization provider ▶ Custom properties ▶ Web attributes (single sign-on) ▶ SSL ▶ Audit ▶ LTPA authentication mechanism ▶ Kerberos authentication mechanism 	<ul style="list-style-type: none"> ▶ Enablement of application security ▶ Java 2 security ▶ User realm (registry) ▶ Trust Association Interceptor (TAI) ▶ SPNEGO Web authentication ▶ RMI/IIOP Security (CSlv2 protocol) ▶ JAAS ▶ Authentication mechanism attributes ▶ Authorization provider ▶ Custom properties

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

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

Security domains can be created and managed by using the administrative console or scripts. Only users who are assigned to the administrator roles can configure security domains. The global configuration data is stored in the `security.xml` file, which is in the `profile_home/cells/cell_name` directory.

For every security domain that is configured. Two files are created in the `profile_name/config/waspolices/default/securitydomains/domain_name` directory:

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

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

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

User account repository

With this option, you can configure the user registry to be used and the authentication settings. You choose to use federated repositories, local repository, or an LDAP repository. WebSphere Application Server uses registries to authenticate users and to retrieve users and groups to create a map of security roles that are used to authorize user actions.

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

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

- ▶ Local operating system (local OS)

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

- ▶ Federated repositories

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

- ▶ Stand-alone LDAP

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

- ▶ Stand-alone custom registry implementation

Custom registries can be written and integrated with WebSphere Application Server. For more information see “Standalone custom registries” in IBM Knowledge Center:

<http://ibm.co/1iPjox3>

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

A user registry provides the application server with user and group information for mapping with Java EE security roles. *Groups* are the method that is used by a registry to represent users who share a common function or attribute.

Several other properties can be set from the secure administration, applications, and infrastructure settings page (shown in Figure 3-2 on page 65). Some of them are used only if administrative security is enabled, such as user account repository, security domains, authentication, and authorization providers.

Authentication

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

The authentication mechanism works with the user registry to verify the identity of the client and then creates a *credential* that represents the authenticated client. The abilities of the credential are determined by the configured authentication mechanism. Only a single active authentication mechanism can be configured within the cell.

These are the available authentication methods:

- ▶ Lightweight Third Party Authentication (LTPA)

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

- ▶ Kerberos (KRB5)

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

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

- ▶ RSA token authentication mechanism

An RSA is used for the administrative agent only.

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

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

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

- ▶ A common security infrastructure that is supported by a single token mechanism, which is the model of LTPA and Kerberos. LTPA is the WebSphere Application Server SSO token, and it is the default SSO mechanism. Kerberos is an industry token for SSO.

- Identity assertion models where systems are configured to trust the authentication that was validated on other systems. This mechanism asserts the user identity without actually supplying the user authentication token (that is, the password). The mechanism relies on a trust between systems. This approach has many forms. Two examples of standards that are supported by WebSphere Application Server are Common Security Interoperability Version 2 (CSIV2) and certain token types that are used in WS-Security.
- You can implement SSO using a re-authentication model. This approach requires passing a user authentication token to the downstream system.

After a user is authenticated, whether via SSO or a password supplied in a web application, the application server authorizes the user to verify that the user of the validated identity is permitted access to the requested resource. An example of the core components involved in authentication for a trust token authentication flow is shown in Figure 3-5. This diagram is to help you consider the authentication and authorization choices that need to be made before and when enabling security.

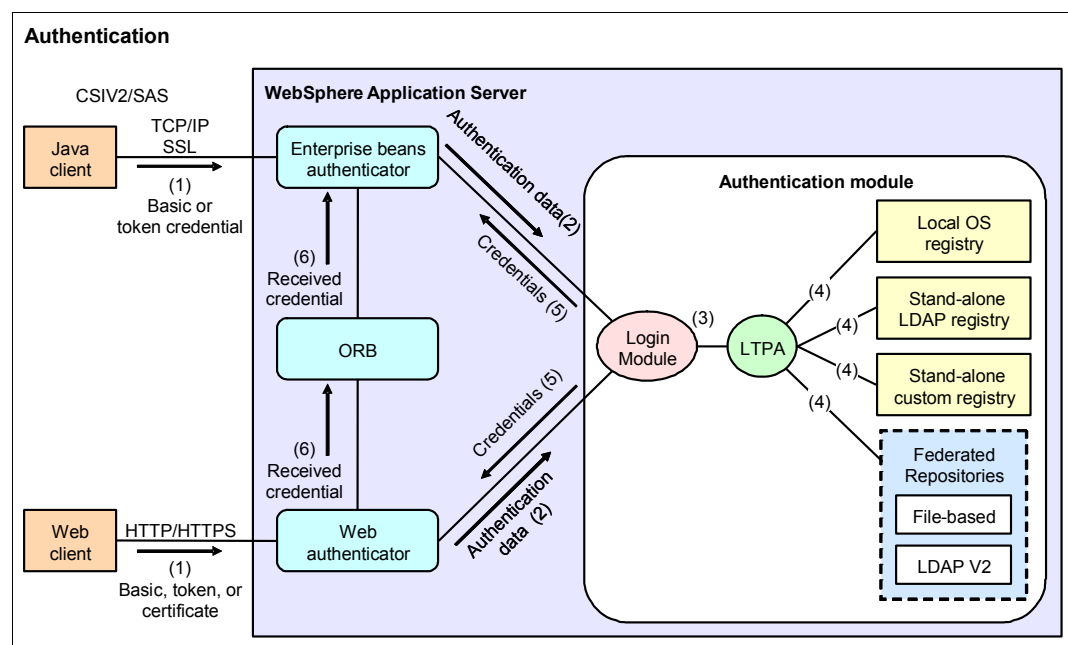


Figure 3-5 WebSphere Application Server simplified authentication component diagram

Figure 3-5 illustrates the following simplified authentication flow:

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

In interpreting the diagram in the Figure 3-5 on page 73, it is useful to identify the security decisions that have been made in relation to authentication and authorization. The following list summarizes the implied and explicit security decisions that were made with regard to Figure 3-5 on page 73:

- ▶ Web requests are participating in a form of SSO, implied by the passing of the token on the request.
Security decision: Enable SSO.
- ▶ The token of choice for SSO is LTPA.
Security decision: Choose an authentication mechanism.
- ▶ The use of a token authentication mechanism indicates establishing a trust domain.
Security decision: Determine the boundaries of the trust.
- ▶ EJB client requests are secured, and EJB clients use CSv2 communications.
Security decision: Secure RMI/IOP communications.
- ▶ SSL communications are used on both RMI/IOP and web-based communications.
Security decision: Secure transports.
- ▶ Identity attributes are propagated between channels.
Security decision: Propagate identity.
- ▶ Having authenticated the request, authorization checking will have been processed. Because there is no indication of an external authorization provider, it can be implied that the default internal authorization provider was used.
Security decision: Use default authorization provider.

Java Authentication and Authorization Service (JAAS)

In rare cases, standard authentication mechanisms might be not sufficient. WebSphere Application Server provides Java Authentication and Authorization Service (JAAS) to allow you to use customized login mechanisms.

Use Java Authentication and Authorization Service (JAAS) login modules when subject modification is required, additional custom authentication is required, or identity mapping is necessary. Use this option for Remote Method Invocation (RMI) authentication. A JAAS login module is also called during propagation login.

Note: For more information, see “Using the Java Authentication and Authorization Service programming model for Web authentication” in the IBM Knowledge Center:

<http://ibm.co/1VEj5XV>

WebSphere Application Server supports many JAAS plug-in points. By using a custom login module, you can make additional authentication decisions or add information to the Subject to make additional, potentially finer-grained, authorization decisions within Java EE application.

The defaults are normally sufficient for most needs. If you are considering customizing login modules, we suggest that you seek IBM or IBM Business Partner guidance to understand the security impacts of this choice.

For more information about developing custom login modules, see “Developing custom login modules for a system login configuration for JAAS” in the IBM Knowledge Center:

<http://ibm.co/1P3Chsp>

3.2.2 SSL/TLS

WebSphere Application Server uses the Secure Sockets Layer (SSL) protocol to provide transport layer security that ensures a secure connection between a client and server. WebSphere Application Server provides a full set of mechanisms for managing SSL configuration.

SSL is the industry standard for data interchange encryption between clients and servers. The foundation technology for SSL is public key cryptography. It uses a combination of asymmetric and symmetric key encryption models.

A series of messages, referred to as a handshake, are exchanged at the start of an SSL session. These messages allow the client to authenticate the server using public key techniques and, optionally, for the server to authenticate the client. The handshake uses asymmetric keys that consist of a public key and a private key. The public key can be distributed widely, but the private key is never distributed; it is always kept secret. When an entity encrypts data using a public key, only entities with the corresponding private key can decrypt that data.

During the handshake, the client and server cooperate in creating symmetric keys for speed and efficiency to use for further communication. Java Secure Sockets Extension (JSSE) is used in WebSphere to handle the handshake negotiation and protection capabilities that are provided by SSL to ensure secure connectivity exists across most protocols. JSSE provides a Java specification for the implementation of the industry standard X.509 standard public key infrastructure (PKI).

A PKI represents a system of digital certificates, certificate authorities, registration authorities, a certificate management service, and a certification path validation algorithm. A PKI verifies the identity and the authority of each party that is involved in an Internet transaction, either financial or operational, with requirements for identity verification. It also supports the use of certificate revocation lists (CRLs), which are lists of revoked certificates.

An SSL configuration determines whether one entity can connect to the other entity and the peer connection that is trusted by an SSL handshake. If you do not have the necessary certificates, the handshake fails because the peer is not trusted.

Figure 3-6 shows how the SSL Conversation happens.

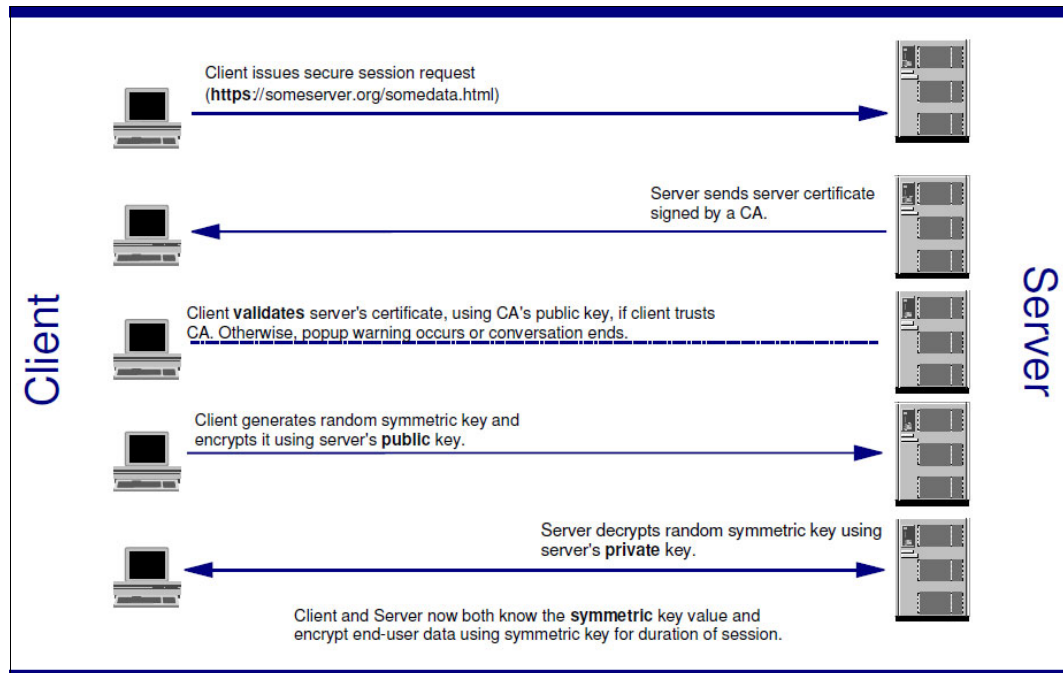


Figure 3-6 SSL Conversation

An SSL configuration determines whether one entity can connect to the other entity and the peer connection that is trusted by an SSL handshake. If you do not have the necessary certificates, the handshake fails because the peer is not trusted. Figure 3-7 shows a hypothetical keystore and truststore configuration example. Private keys are stored in the keystore file. The public keys are stored in the form of trusted certificates in the truststore.

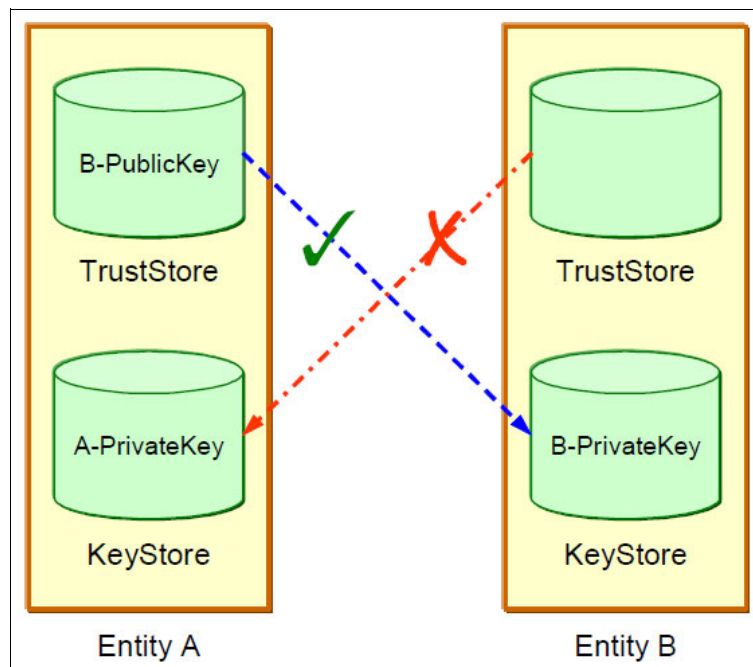


Figure 3-7 Keystore and TrustStore

The truststore for Entity A contains the public key of Entity B. Entity A can connect to Entity B, because Entity B can decrypt the data using its private key, but the truststore for Entity B does not contain the public key of Entity A. Therefore, if Entity A requires validation with Entity B, the handshake will fail.

You can think of Entity A as a client and Entity B as a server. If SSL is configured for server authentication, it will work in this scenario. But if SSL is configured for both client and server authentication, it will fail because there is no trust key on the server side to authenticate the client.

SSL can still be widely used on WebSphere Application Server to enforce security. It can be enabled with LDAP communications, IBM MQ, etc.

3.2.3 Java security

Java EE applications are built from components and associated parts. Components are servlets and JavaServer Pages (JSPs), collectively called web components or sometimes web applications. There are also Enterprise JavaBeans (EJBs) or EJB components.

JSPs contain standard HTML tags and are similar in structure to HTML pages. They also contain Java programming language statements, which allow the Web application programmer to create pages dynamically.

JSPs are not compiled by the application developer, but installed in source format. They are compiled when the application executes. When compiled, JSPs appear to the runtime to be identical to servlets.

Servlets are similar in concept to JSPs, but structured differently. Primarily, servlets contain Java programming language statements, but may also have embedded HTML tags and associated text. An application programmer writes and compiles servlets, and the compiled byte codes are installed with the Java EE application.

The notion of security includes several aspects. *Identification* is the process by which users or programs that communicate with each other provide identifying information. *Authentication* is the process used to validate the identity information provided by the communication partner. *Authorization* is the process by which a program determines whether a given identity is permitted to access a resource, such as a file or application component. The Java EE specifications describe the concepts to be used for these processes.

Although data integrity, confidentiality, non-repudiation and auditing are also important aspects of security, the Java EE specifications do not address them in any detail.

Java EE specifies a component programming model for security that provides both application programming interfaces (APIs) and declared properties. The security APIs used in the logic of application programs are referred to as *programmantic security*. The declared security properties, called *declarative security*, are found in components' deployment descriptors. One objective of the Java EE programming model is to encourage the use of declarative security, which is enforced by the container. This removes much of the responsibility for security from the application developer.

For both programmantic and declarative security, Java EE uses the concept of security roles. Security roles are used to delineate permissions based on business rules.

A role-based model may be contrasted with a more traditional security model, sometimes called permission-based. In a traditional security 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. For example, a user or group might have permission to read a file, but not to update it. Another group might have permission to execute a file (presumably containing program code), but not to browse it with an editor so as to discover its inner logic.

Typically, both the user registry and the resources are defined in a security database. Users and groups are associated with resources in some way, such as with an access control list (ACL). The maintenance of the user registry, resource profiles, and ACLs requires a security administrator who must have enough information about each user and each resource to make the proper decisions. In the Java EE role-based model, users and groups of users are still stored in a user registry. A mapping must also be provided to specify the assignment of users and groups to security roles. This can also be contained in a user registry, or it can exist in a separate file. Java EE applications, however, have their own role-based security constraints, so that the applications themselves do not have to be defined as protected resources by a security administrator or defined in a registry or security database.

Java EE applications, moreover, display considerably more flexibility than traditional applications with respect to security constraints. Typically, applications as a whole are protected, if at all, by an ACL that is global with respect to the program. A user or group of users may or may not have permission to execute the program as a whole, but there is no means of restricting parts of the program short of including authorization logic within the application code itself. Java EE applications have a declarative means of protecting each individual method of each component by specifying, outside the program logic, which security roles may execute it.

To implement both programmatic and declarative security, application developers (including what Java EE refers to as “component providers” and “assemblers”) declare roles names in the component’s deployment descriptor. In addition, for declarative security, the application assembler associates roles with “method-permission” elements in the deployment descriptor.

Deployers make Java EE components and applications ready for particular operational environments. Part of this deployment process includes editing the deployment descriptor (usually through a graphical tool) to map security roles to users or user groups that are understood by the security implementation of the environment.

As illustrated in Figure 3-8 on page 79, application developers (component providers) may use Java EE roles both programmatically and declaratively. The application assembler can link the role names used by application components from multiple sources. The security administrator or deployer provides role names that meet the requirement of the system where the application is being deployed. The security administrator links or permits individual users or groups to the role names.

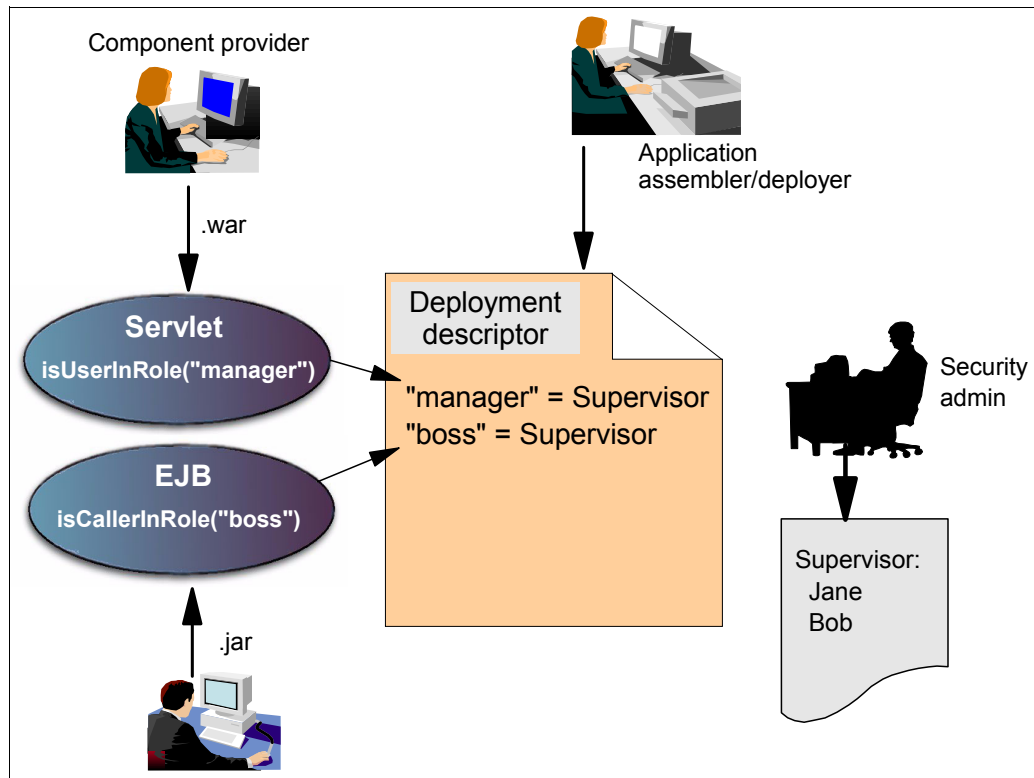


Figure 3-8 Java EE security roles

In the example Example 3-1, the application assembler has linked the role names "manager" and "boss" to the role name Supervisor. The role name declarations and links appear in the deployment descriptor as shown below. It can be edited using WSAD, the Application Assembly Tool (AAT), or other tools.

Example 3-1 Deployment descriptor: Security-role example

```
<servlet>
  <servlet-name>secureEJBCaller</servlet-name>
  <display-name>secureEJBCaller</display-name>
  <servlet-class>ejbTester.secureEJBCaller</servlet-class>
  <security-role-ref>
    <role-name>Laborer</role-name>
    <role-link>Worker</role-link>
  </security-role-ref>
</servlet>
```

The procedures used by the security administrator and deployer vary depending on the deployment platform.

3.3 Interfaces (transaction systems, databases, IBM MQ, web server, and other adapters)

In this section, we will cover some options to enhance security over the middleware that do some kind of interface with WebSphere Application Server, such as WebSphere Message Queue (WebSphere MQ) and DB2.

3.3.1 WebSphere Message Queue

In this section, we will cover some details regarding security over the use of WebSphere Message Queue (WebSphere MQ) as messaging provider in WebSphere Application Server.

WebSphere MQ is a messaging-oriented middleware solution that allows independent and non-concurrent applications on distributed systems to communicate to each other. It assures one time message delivery, and works with many vendor platforms using consistent API.

WebSphere Application Server supports the following messaging providers:

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

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 Java Message System (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.

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

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

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

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

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

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

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

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

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

In each case, SSL can be used to secure communications between WebSphere Application Server and WebSphere MQ, and we suggest that you establish mutual SSL communication between WebSphere Application Server and WebSphere MQ.

The following steps describe how to configure SSL for WebSphere MQ:

1. Enable SSL on WebSphere MQ.
2. Obtain the public certificate for WebSphere MQ from the WebSphere MQ administrator.
3. Create a keystore at the appropriate scope. Choose a scope that will allow all servers that have to connect to WebSphere MQ to have access to the keystore.
4. Import the obtained certificate into the keystore as a signer certificate.
5. In the keystore, create a new personal certificate for authentication to WebSphere MQ.
6. Extract the public part of the new certificate and provide it to the WebSphere MQ administrator, because it must be added to the WebSphere MQ truststore.
7. Create a new SSL configuration at the same scope. Select the new keystore as the keystore and truststore.
8. Configure the JMS WebSphere MQ connection factory in WebSphere:
 - a. Set the correct server connection channel.
 - b. Enable SSL and select the new SSL configuration.
 - c. In the Client transport properties, you can set the peer name that will be checked against the distinguished name (DN) of the WebSphere MQ certificate.

Now, if WebSphere MQ is configured correctly, it will allow only connections that are using SSL from parties that have the correct certificate. For a more detailed explanation about how to execute this procedure, see the IBM Redbooks publication titled *WebSphere Application Server V7.0 Security Guide*, SG24-7660.

Tip: For more information about WebSphere MQ configuration, see the IBM Redbooks publication titled *WebSphere Application Server V7 Messaging Administration Guide*, SG24-7770.

3.3.2 Event monitoring and recording (SMF, internal logging)

You have the option to enable application server and z/OS thread identity synchronization.

This option significantly increases the number of SMF 80 records that are used for security auditing. When security auditing is turned on for SMF 80 records, the amount of DASD used increases significantly.

3.4 Guiding principles for configuring security

This section describes WebSphere Application Server common misconfigurations and security considerations.

3.4.1 Common misconfigurations

These are the most common misconfigurations for WebSphere Application Server for z/OS:

- ▶ Although the command to start the application server is still the same when administrative security is enabled, stopping the server requires extra information. You must specify a user ID with administrator role rights, or else the primary administrative user name specified in the user account repository and its password, in the **stopServer** command:

```
<WebSphere_home>\bin\stopServer.bat <server_name> -username <userID> -password <password>
```

For WebSphere Application Server running under a UNIX-based operating system (OS), the UNIX equivalent of this command carries a serious security problem. Anyone who uses the **ps -ef** command while the stopServer process is running can see the user ID and the password.

To avoid that problem, if you are using the SOAP connection type (default) to stop the server, edit this file:

```
<WebSphere_home>\profiles\<profilePath>\properties\soap.client.props
```

Then, change the values of the following properties:

```
com.ibm.SOAP.securityEnabled=true
```

```
com.ibm.SOAP.loginUserId=<user ID>
```

```
com.ibm.SOAP.loginPassword=<password>
```

- ▶ One of the most common problems that occurs when application security is enabled is that the `getRemoteUser()` method or `getUserPrincipal()` method of the `HttpServletRequest` interface returns a null value. This happens if, for example, authentication is done in the HTTP Server container before reaching the WebSphere Application Server. Whenever application security is enabled, WebSphere Application Server passes the authentication token only to *secure* resources within its container.

To secure these resources, add a security constraint within the `web.xml` application descriptor file, as shown in Example 3-2.

Example 3-2 Securing the resource `/securedhello` URI

```
<security-constraint>
  <display-name>Authenticated</display-name>
  <web-resource-collection>
    <web-resource-name>Authenticated Resources</web-resource-name>
    <url-pattern>/securedhello</url-pattern>
    <http-method>PUT</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <description>Authorized guest roles</description>
    <role-name>ServletGuest</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>INTEGRAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<security-role>
  <description>Authenticated guest for servlet</description>
  <role-name>ServletGuest</role-name>
</security-role>
```

Remember to define a correct security role mapping for the role that you have added when deploying your application. In Example 3-2, the name of the role is *ServletGuest*.

3.4.2 Security considerations

This section lists considerations and practices to follow during configuration of your WebSphere Application Server for z/OS:

- Preferred practices for keystores

Keystores can hold multiple key and certificate entries in one file. WebSphere distinguishes two different keystores when configuring security:

- Keystore

- You keep your private keys in the keystore. These keys are either self-signed or issued by a certificate authority.

- Truststore

- The truststore keeps the public key of your own key pair and the public certificates of the parties you trust. It is wise to keep the two different keystores and keys and certificates separate.

After you have issued or received your own key pair, store the private key locked with a password and never open it again. The truststore might be changed occasionally to add a new certificate that you trust.

- Preferred practice for Java 2 security

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

- ▶ Wsadmin will go through a firewall if you are using SOAP and correct ports are opened.
- ▶ Take additional steps to further protect against internal network access:
 - Consider limiting access to Administrative functions such as the Web Administration Console to trusted VLANs or an internal VPN.
 - Consider a full internal DMZ.
 - Follow preferred practices for security configuration.
 - Configure WebSphere Application Server security.
- ▶ This tip describes the steps to set up the security configuration of the IBM WebSphere Applications server for running a sample application. The sample application demonstrates how dependencies in the Distributed Computing Environment (DCE) can be replaced by using, for example, WebSphere Application Server. The components to be configured are described in more detail in the IBM Redbooks publication titled *WebSphere Application Server V8.5 Administration and Configuration Guide for the Full Profile*, SG24-8056:

<http://www.redbooks.ibm.com/abstracts/sg248056.html>

Follow these steps:

- a. Prepare SSL key and trust files. These files are needed for the SSL configuration.
 - b. Configure SSL. This creates an SSL configuration to be later assigned to the CSiv2 communication mechanism.
 - c. Configure CSiv2. WebSphere Application Server provides a communication mechanism with CSiv2 to pass the authentication information of the JEE client to the enterprise application.
 - d. Configure LDAP as the user registry. Because an LDAP directory is used in the sample application as a user registry, the LDAP server access is configured in this step.
 - e. Configure LDAP filter rules. The configuration of the LDAP filter rules is necessary to match client certificate information to DCE users (principals) in the LDAP directory.
 - f. Configure LTPA as the authentication mechanism. This is to specify the LTPA facility as an authentication mechanism for the IBM WebSphere Application Server.
- ▶ Each of the elements must be configured to establish a security framework in which the scenario application is secured at the same level as an application is in a DCE environment. Assuming that authorization is done within the enterprise application, then authentication, authorization, and encryption are covered by the security configuration of the IBM WebSphere Application Server.
 - ▶ With the possible exception of z/OS, LDAP directories are the repository of choice. While repository types other than LDAP are supported, only LDAP is the recognized industry standard.
 - ▶ When designing the security access model for your application server cell, we recommend that all access relationships are mapped to groups in the registry in preference to individual users, even if the group only has one user. This approach decouples individual users from applications and administrative functions in the cell, making security administration significantly more manageable.

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

There are three fundamental principles of a DMZ that need to be considered here:

- Inbound network traffic from outside must be terminated in the DMZ. A network transparent load balancer, such as Network Dispatcher, doesn't meet that requirement alone.
- The type of traffic and number of open ports from the DMZ to the intranet must be limited.
- Components running in the DMZ must be hardened and follow the principle of least function and low complexity.

Therefore, it is normal to place the Web server in the DMZ and the WebSphere Application Server application servers inside the inner firewall. This is ideal, as the Web server machine can then have a very simple configuration and require very little software. It also serves as a point in the DMZ that terminates inbound requests. And finally, the only port that must be opened on the inner firewall is the HTTP(S) port for the target application servers. These steps make the DMZ a very hostile place for an attacker. It is also appropriate to put a secure proxy server in the DMZ instead of or in addition to the Web server if that is preferred.

If you place WebSphere Application Server on a machine in the DMZ, then far more software must be installed on those machines and more ports must be opened on the inner firewall so that WebSphere Application Server can access the production network. This largely undermines the value of the DMZ.

You may choose to place something other than a Web server in the DMZ. It is also reasonable to put a secure proxy into the DMZ, such as IBM Tivoli Access Manager WebSEAL, the WebSphere Application Server V7 secure proxy, or a hardened appliance such as IBM WebSphere DataPower®. The key is that what you put into the DMZ must NOT be a complex application server, must be hardened, and must terminate inbound connections.

For more information, see “Configuring a DMZ Secure Proxy Server for IBM WebSphere Application Server using the administrative console” in the IBM Knowledge Center:

<http://ibm.co/1PTb78A>

- Separate your production network from your intranet

Most organizations today understand the value of a DMZ that separates the outsiders on the Internet from the intranet. However, far too many organizations fail to realize that many intruders are on the inside. You need to protect against internal as well as external threats. Just as you protect yourself against the large untrusted Internet, you should also protect your production systems from the large and untrustworthy intranet.

Separate your production networks from your internal network using firewalls. These firewalls, while likely more permissive than the Internet-facing firewalls, can still block numerous forms of attack. After applying this step and the previous step, you should end up with a firewall topology like the one shown in Figure 3-9.

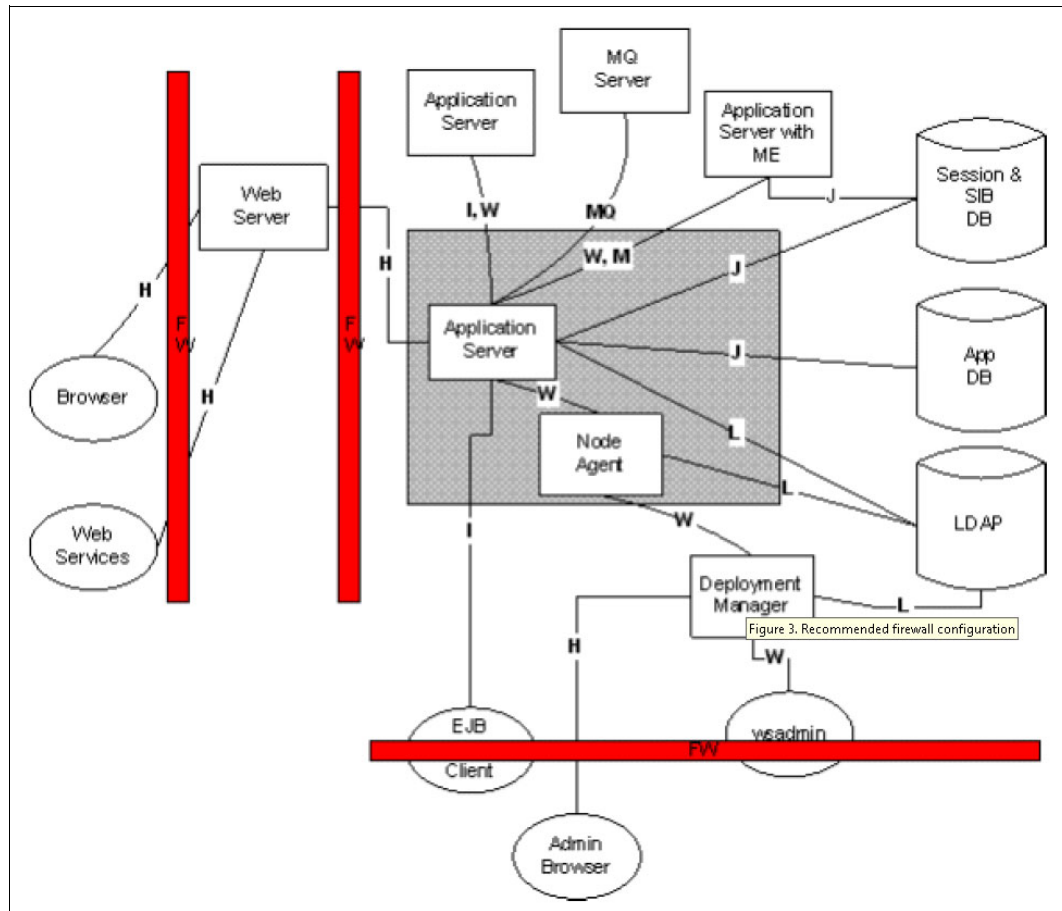


Figure 3-9 Suggested firewall configuration for WebSphere Application Server

Notice that wsadmin has been placed on the edge of the firewall. This is attempting to show that while it is preferred that wsadmin be run only within the production network (within protected area), wsadmin access can also be limited to selected addresses, corresponding to administrator desktops, fairly easily through a firewall. You can see in Figure 3-9 that EJB clients are also on the edge, as they might be on either side of the firewall.

A single firewall and not a full DMZ facing the intranet is shown here, as this is the most common topology. However, we increasingly see full DMZs (with a Web server in the internal DMZ) protecting the production network from the non-production intranet. That is certainly a reasonable approach.

- Do not put the ODR in the DMZ

Before V8.5, WebSphere Application Server V8.5 includes the on-demand router (ODR), previously only available on WebSphere eXtreme Scale. If you are using the on-demand router function (ODR), you should proceed in a different way. The secure topology for using ODR is shown in Figure 3-10.

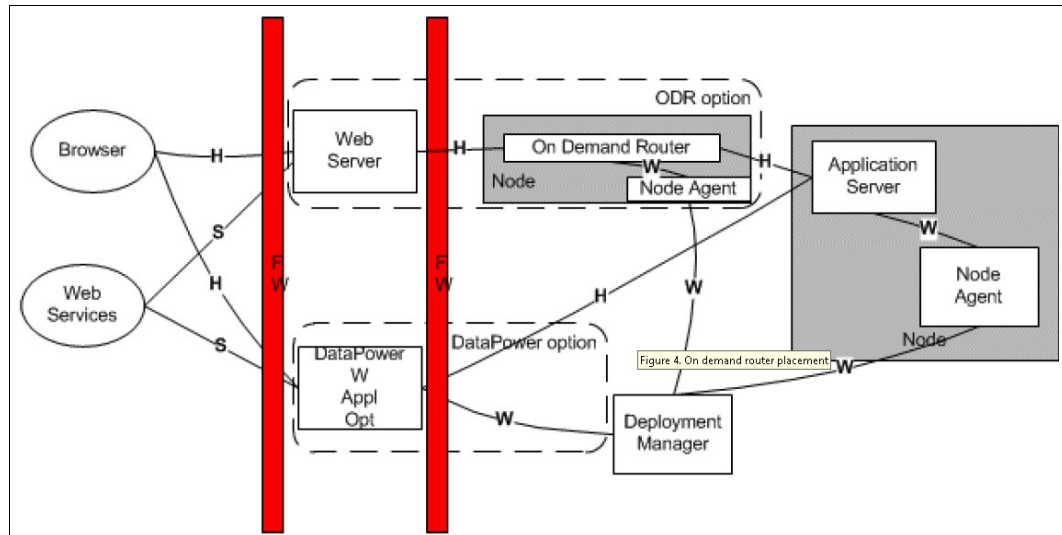


Figure 3-10 On-demand router placement

Figure 3-10 shows the placement of the Java-based ODR server, and contrasts that with the on demand routing abilities available with the Application Optimization (AO) feature of the IBM DataPower SOA Appliances. While the Java ODR is not placed in the DMZ, the DataPower SOA appliance is a hardened device suitable for placement in the DMZ.

- Use HTTPs from the browser

If your site performs any authentication or has any activities that should be protected, use HTTPS from the browser to the Web server. If HTTPS is not used, information such as passwords, user activities, WebSphere Application Server session cookies, and LTPA (see sidebar) security cookies can potentially be seen by intruders as the traffic travels over the external network.

For applications that enable HTTP traffic before authentication, make sure you pay close attention to cookies. If a cookie (such as the JSESSIONID) is set before HTTPS being used, that cookie is a potential risk after HTTPS is used because it might have been altered or captured by an intruder. This is why WebSphere Application Server has a separate cookie for authenticated users. An even more subtle attack is that any page returned over HTTP can be potentially altered by an intruder, even URLs embedded in the page. Therefore, a user might click a “secure” URL on your page but be directed to an intruder’s site.

- Keep up to date with patches and fixes

As with any complex product, IBM occasionally finds and fixes security bugs in WebSphere Application Server, IBM HTTP Server, and other products. It is crucial that you keep up to date on these fixes. It is important that you subscribe to support bulletins for the products you use and, in the case of WebSphere Application Server, monitor the security bulletin site for your version:

Security Bulletin for WebSphere Application Server

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg21368398>

These bulletins often contain notices for recently discovered security bugs and the fixes. You can be certain that potential intruders learn of those security holes quickly. The sooner you act the better.

- Restrict access to WebSphere MQ messaging

If you are using WebSphere MQ as your messaging provider, you'll need to address queue authorization via other techniques. WebSphere MQ by default does not perform any user authentication when using client/server mode (bindings mode relies on process to process authentication within a machine). In fact, when you specify a user ID and password on the connection factory for WebSphere MQ, those values are simply being ignored by WebSphere MQ.

One option to address this is to implement your own custom WebSphere MQ authentication plug-in on the WebSphere MQ server side to validate the user ID and password sent by WebSphere Application Server. A second, and likely simpler technique is to configure WebSphere MQ to use SSL with client authentication and then ensure that only the WebSphere Application Server server possesses acceptable certificates for connecting to WebSphere MQ.

- Web authentication trust risk

The mechanism used for validating certificates must be configured securely; by default, it is not for Web traffic. When certificate-based authentication is used for Web authentication, a very subtle potential trust issue occurs. When a web client authenticates to the web server, the web server validates the certificate. Then, the WebSphere Application Server Web server plug-in forwards the certificate information from the Web server to the application server. This information is forwarded so that the Web container can map that certificate to a Java EE identity. The problem is that the information is just a description of the certificate (information that is available in the public certificate). If an intruder can connect directly to the Web container and bypass the Web server, the system is now vulnerable to compromise because the intruder could forge certificate information and trick the runtime, enabling them to become anyone. This means that if you are using certificates for authentication (Java EE-based authentication or custom application code directly examining the certificate), you must block the vulnerability.

If your intention is to use certificates to authenticate directly to the Web container (meaning there is no Web server in your scenario), you'll have to configure the Web container to ignore certificate information in the HTTP headers (in this scenario, such information would always be a forgery). To do this, you must configure the "trusted" custom property on the Web container for each application server and set its value to false, as shown in Figure 3-11.

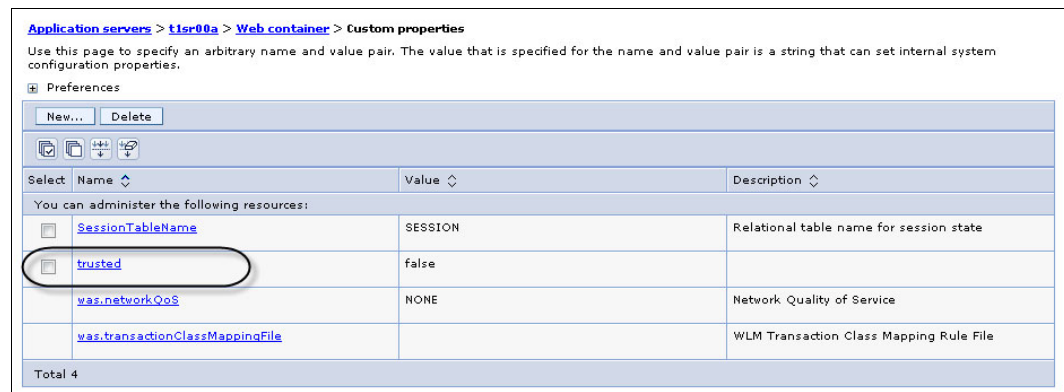


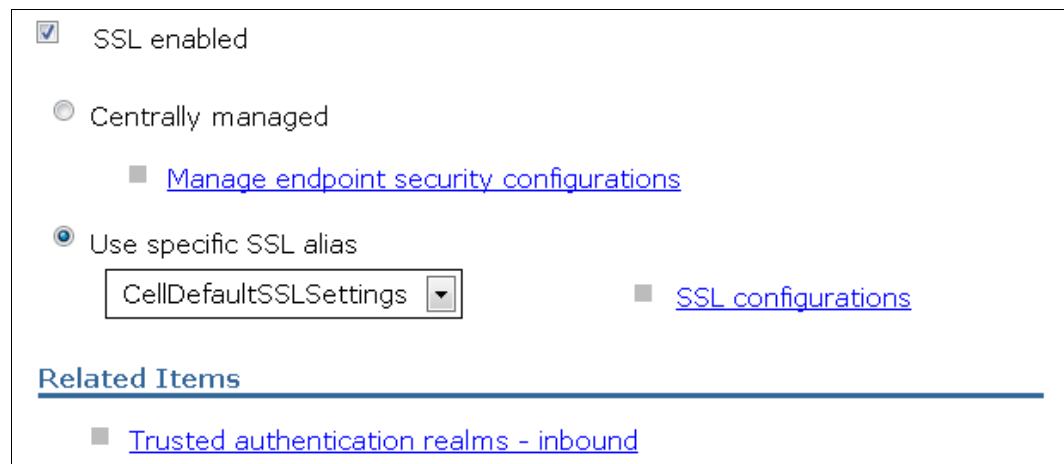
Figure 3-11 Setting web container to ignore certificates headers from client

- Protect configuration files and private keys

Limit file system access to WebSphere Application Server's files by using operating system file permissions. WebSphere Application Server, like any complex system, uses and maintains a great deal of sensitive information. In general, almost no one should have read or write access to most of the WebSphere Application Server information. In particular, the WebSphere Application Server configuration files (<root>/config) contain configuration information as well as passwords.

- Encrypt WebSphere Application Server to LDAP link

When using an LDAP registry, WebSphere Application Server verifies a user's password using the standard `ldap_bind`. This requires that WebSphere Application Server send the user's password to the LDAP server. If that request is not encrypted, a hacker could use a network sniffer to steal passwords of users authenticating (including administrative passwords). Most LDAP directories support LDAP over SSL, and WebSphere Application Server can be configured to use this. As Figure 3-12 shows, you can configure it on the LDAP user registry panel, checking the use SSL enabled option and then configuring an SSL configuration appropriate to your LDAP directory. You'll most likely need to place the signing key for the LDAP server's certificate into the truststore. It's best to create a new SSL configuration just for LDAP to avoid causing problems with the existing SSL usage.



☒ SSL enabled
☐ Centrally managed
 ■ [Manage endpoint security configurations](#)
☒ Use specific SSL alias
 CellDefaultSSLSettings ▼ ■ [SSL configurations](#)

Related Items
 ■ [Trusted authentication realms - inbound](#)

Figure 3-12 Enable LDAP SSL

If you use a custom registry, you'll obviously want to secure this traffic using whatever mechanisms are available.

- Do not specify passwords on the command line

Once security is enabled, the WebSphere Application Server administrative tools require that you authenticate in order to function. The obvious way that you would think to do this is to specify the user ID and password on the command line as parameters to the tools.

Do not do this.

This exposes your administrative password to anyone looking over your shoulder. And, on many operating systems, anyone that can see a list of processes can see the arguments on the command line. Also, previous commands are visible in the command history, including those password arguments.

Instead, ensure that the administrative tools prompt for a user ID and password. In all currently supported versions of WebSphere Application Server, this is the default, so no explicit actions are needed to ensure that this occurs.

If you find it annoying that the command line tools prompt graphically for a user ID and password, you can override this behavior and force the tools to use a simple text-based prompt. To do this, you must change the `loginSource` from `prompt` to `stdin` by editing the appropriate configuration file. By default, the administrative tools use SOAP, so the `soap.client.props` file should be edited. If you are using RMI, edit `sas.client.props`. Look for the `loginSource` property in the appropriate file and change it to specify `stdin`.

- Consider using more salt in file-based federated repository registry

If you are using the Federated Repository registry and are also using the built-in file repository in that registry, the users in that registry have their user IDs and passwords stored in the `fileregistry.xml` file in the cell config directory. These passwords are one-way hashed, which means that you cannot derive the actual password from the hashed value. The hashes are computed with cryptographic salt.

In WebSphere Application Server V8.0 and 8.5, it is possible to specify the length of that salt and the hashing algorithm that is used. This file should be protected by OS-defined access control lists so that only privileged OS users are able to read the file. To prevent password cracking attacks via rainbow tables, consider increasing the amount of salt or using a stronger hashing function.



Transaction processing systems

In this chapter, we discuss security concepts and provide an architectural overview of transaction processing systems on IBM z Systems. Then, we describe guiding principles for configuring transaction processing system security.

This chapter covers the following topics:

- ▶ IBM CICS Transaction Server
- ▶ IBM Information Management System Transaction Manager

4.1 IBM CICS Transaction Server

Modern enterprise systems typically consist of multiple software components, for example, IBM WebSphere Application Server running on a distributed platform connecting to a back-end IBM CICS Transaction Server (TS) running on IBM z/OS operating system. Another way to use CICS is as a front-end system using IBM 3270 terminals and emulators to invoking IBM CICS transactions. CICS transactions can also be initiated from mobile, web, and other transaction processing environments. Because of all these numerous methods for initiating transactions in CICS, discussing security considerations for all of them in a single chapter is difficult. Therefore, we focus on the most commonly used methods that fall into one of the following three categories:

- ▶ 3270 terminal-based transactions

These are traditional CICS transactions that are initiated from a 3270 terminal by a user.

- ▶ CICS intercommunication

These transactions originate from other CICS or non-CICS systems using a similar communication facility. This includes multiregion operation (MRO) links, intersystem communication (ISC) over SNA, and IP intercommunication (IPIC).

- ▶ Distributed systems

These include transactions initiated from a distributed environment where CICS is the service provider. We focus on web services, web support, IBM MQ integration, and TCP/IP sockets.

For information about security for CICS Transaction Gateway, see the IBM Redbooks publication titled *The Complete Guide to CICS Transaction Gateway Volume 1 Configuration and Administration*, SG24-8160.

4.1.1 Security concepts and architecture

First, let's briefly review some of the important security concepts in a transaction processing system:

- ▶ **Authentication**

CICS validates the identity claimed by the user or entity invoking the transaction. Identity can be in the form of user ID and password from a terminal or passed along with the request for CICS intercommunication. The identity can also be a token that is provided by a trusted party, such as a Security Assertion Markup Language (SAML) assertion or an X.509 certificate.

- ▶ **Identification**

Following authentication CICS assigns the identity to the user or entity invoking the transaction. The identity is typically an external security manager (ESM) ID, such as a z/OS RACF Security Server user ID.

- ▶ **Authorization**

CICS uses the z/OS System Authorization Facility (SAF) interface to check access to the resources requested by the user transaction.

- ▶ **Integrity**

Integrity is the proof that transmitted and stored information cannot be intentionally or accidentally altered in any manner. CICS ensures integrity by using encryption for transmitted data and security controls to prevent unauthorized access to the user data.

- ▶ **Confidentiality**

A confidential message is one that can only be deciphered by the parties who are authorized to receive them. This is achieved in CICS using different encryption schemes. Proper encryption ensures that unauthorized parties cannot decrypt to the original message.

- ▶ **Auditing**

CICS can be configured to capture and record security-related events (such as a user signing on or off the system) so that they can be analyzed later.

- ▶ **Non-repudiation**

Non-repudiation is the ability to provide proof that the sender sent a message and that the message is identical to the one received. This can be achieved in CICS by using digital certificates and a public key infrastructure (PKI).

Next, we review how these security concepts apply to CICS environment by addressing the following topics in the subsections that follow:

- ▶ Traditional CICS authentication
- ▶ CICS intercommunication authentication
- ▶ CICS web services authentication
- ▶ CICS web support authentication
- ▶ IBM MQ authentication
- ▶ CICS TCP/IP sockets authentication
- ▶ Authorization
- ▶ CICS DB2 interface
- ▶ CICS IMS interface
- ▶ Integrity and confidentiality
- ▶ Auditing and monitoring
- ▶ Non-repudiation

Traditional CICS authentication

Authentication is usually the first step in a security request workflow. After the user is authenticated, an identity can be asserted to the downstream process steps.

Signon through terminal

CICS can be configured to ask terminal users to sign on, as shown in Figure 4-1. CICS users can use the CICS-supplied transaction CESN or can use their own custom signon transaction that uses a **SIGNON** command. During signon, CICS issues a request to an ESM, such as RACF, to authenticate a user's password.

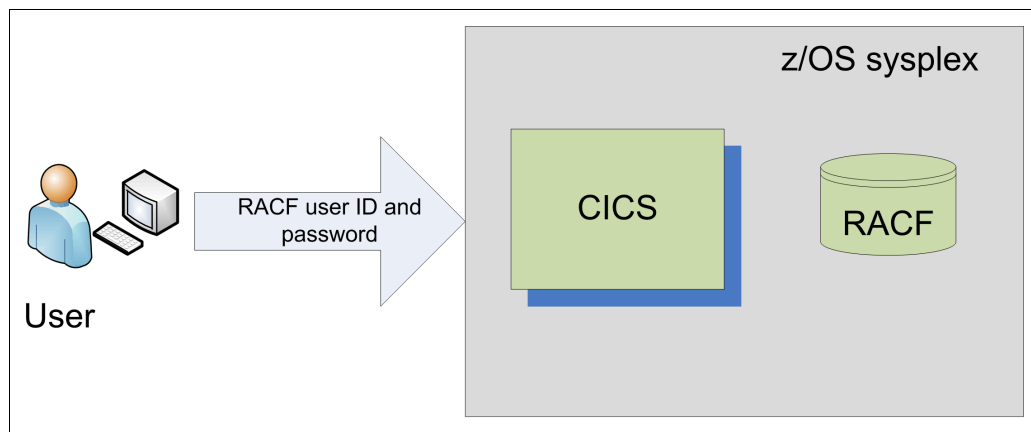


Figure 4-1 Traditional CICS signon

CICS also allows *preset terminal* security. This feature enables an administrator to associate a terminal with a user ID. This is normally done for printers and other devices that do not have a keyboard, but it can also be applied to any physical terminal. Be sure to take proper care before using preset terminal security, because anybody who has physical access to the terminal can issue transactions that are authorized for the associated user ID of the terminal.

CICS special user IDs

CICS uses two particular user IDs in addition to those that identify individual users:

- | | |
|------------------------|--|
| Region user ID | The CICS region user ID is used for checking the authorization when the CICS system, rather than an individual user of the system, requests access to system resources, such as CICS data sets and other servers. |
| Default user ID | When a user does not sign in, CICS assigns a default user ID to the user. It is specified in the system initialization table (SIT) parameter DFLTUSER. In the absence of more explicit identification, it identifies TCP/IP clients that connect to CICS, providing little authority to the default user ID. |

CICS intercommunication authentication

So far, we've looked at CICS as a single system. However, CICS can also be used in a multi-system environment in which it can communicate with other systems that have similar communication facilities. This is called *CICS intercommunication*. It is the communication between a local CICS system and a remote system, which might or might not be another CICS system. Here, we describe security considerations for such multi-system intercommunication, which could be one of the following types:

- ▶ Multiregion operation (MRO)
- ▶ Advanced program-to-program communication (APPC) or LUTYPE6.2
- ▶ LUTYPE6.1
- ▶ IP interconnectivity (IPIC)

Now we explain the authentication mechanism for each of these CICS intercommunication types. For CICS, intercommunication authentication is done in two parts:

- ▶ Bind-time security
- ▶ Link-time security

Bind-time security

Authentication of clients connecting to CICS using CICS intercommunication mechanisms are done during establishment of the communication session. This level of security is often called bind-time security or session security.

Figure 4-2 depicts bind-time security for the various CICS intercommunication types.

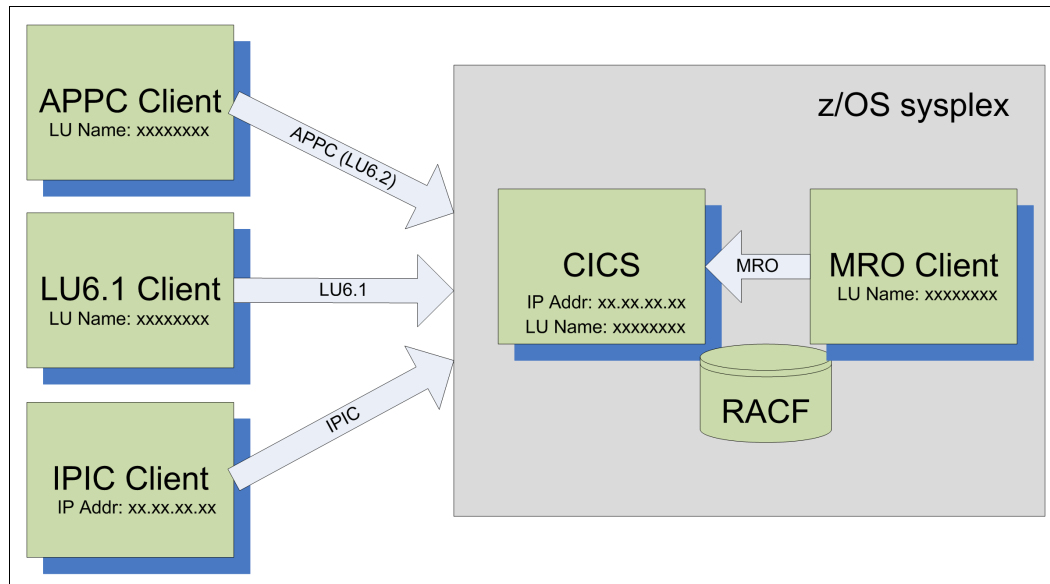


Figure 4-2 CICS intercommunication authentication

Bind-time security is achieved in following ways for different types of CICS intercommunication:

- MRO connection

Clients connecting to a server CICS region using MRO should reside on the same z/OS sysplex. As a result, they use the same security manager as the server CICS region. This simplifies authentication and identification steps. To set up MRO, the client user ID for the connection is required to be added to the DFHAPPL.applid RACF FACILITY class profile of the server CICS region. This security check is mandatory for MRO connections.

- APPC or LU6.2 connection

Clients from a remote SNA network can use APPC or LUTYPE6.2 (or simply LU6.2) connections to access CICS region. To secure such connections, bind-time security is used. Although, bind-time security is optional for LU 6.2 connections, it is a good practice to set up. Authentication for bind-time security is done only during establishment of session and is not repeated as long as the session is active, so it does not cause performance delays.

To configure bind-time security, ensure that BINDSECURITY(YES) is specified in the CICS System Definition (CSD) connection definition and that SEC=YES and XAPPC=YES parameters are set in the CICS system initialization table (SIT). If either of these is not set, that leads to skipping bind security for the LU6.2 connection.

In addition, it is advisable to add a RACF profile under APPCLU class for each client by using the following command where the local LU is lnetwork-id.local-lu-name and the remote LU is rnetwork-id.remote-lu-name. For example:

```
RDEFINE APPCLU lnetwork-id.local-lu-name.rnetwork-id.remote-lu-name UACC(NONE)
```

- ▶ LUTYPE6.1 connection

Clients from a remote SNA network can also use older LUTYPE6.1 (or simply LU6.1) connections. However LU6.1 connections do not use bind-time security. As a good practice, your CICS region should only allow such connection if the remote client is trusted.

- ▶ IPIC connection

An IPIC connection is similar to using LU6.2 or LU6.1, with the difference that the connection is over IP network rather than SNA network. Remote clients for IPIC are defined using the TCPIP SERVICE resource definition in local CICS CSD.

For IPIC connections, bind security is implemented by using a client-authenticated SSL connection. In this configuration the client application needs to be authenticated by the CICS server before they are allowed to successfully connect.

Link-time security

Link-time security checking is attempted when CICS attaches the invoked transaction to a user ID. CICS intercommunication methods attach a link user ID to the transaction. Link user ID is used to check access of the transaction to various CICS resources. Link user ID is set based on the ATTACHSEC parameter of the connection definition:

- ▶ ATTACHSEC=LOCAL

A user identifier is not expected from the client. CICS uses the local user profile for the transaction.

- ▶ ATTACHSEC=IDENTITY

The client is required to provide a user identifier for link request. The user must be defined to local RACF.

- ▶ ATTACHSEC=VERIFY

The client is required to provide a user identifier and password for link request. The user must be defined to local RACF.

Note: In cases where link user ID is not valid or revoked, CICS attaches the default CICS ID (as defined in the DFLTUSER system initialization parameter) to the transaction.

For more information about CICS intercommunication, see “CS Transaction Server for z/OS Version 5 Release 1” in the IBM Knowledge Center:

<http://ibm.co/1VpYW2b>

CICS web services authentication

The web services protocol is the most common protocol that is used to build service-oriented architecture (SOA) applications. A service is an application component that has a well-defined published interface that allows other application components to invoke operations on the service without any knowledge of how the service is implemented. The web services specification is based on open standards, such as Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP). CICS support for web services conforms to following standards:

- ▶ SOAP 1.1 and 1.2
- ▶ Hyper Text Transfer Protocol (HTTP) 1.1
- ▶ Web Services Descriptor Language (WSDL) 1.1 and 2.0

Use of open standards is good for interoperability in a heterogeneous environment with a large number of clients gaining access to application programs running in CICS, but it also opens new security challenges in securing CICS. In the subsections that follow, we describe some of the methods to secure a CICS region and application programs participating in a web services environment.

Transport security

Web services in CICS can be invoked by using HTTP or IBM MQ. See “CICS web support authentication” on page 99 to configure transport security for web services over HTTP, and see “IBM MQ authentication” on page 100 for web services over IBM MQ.

Message security

Complex distributed systems can have one or more intermediary servers in between the web services requester and CICS that is acting as the web services provider, as shown in Figure 4-3. Message-based security can be used to secure the SOAP messages in such complex system configurations.

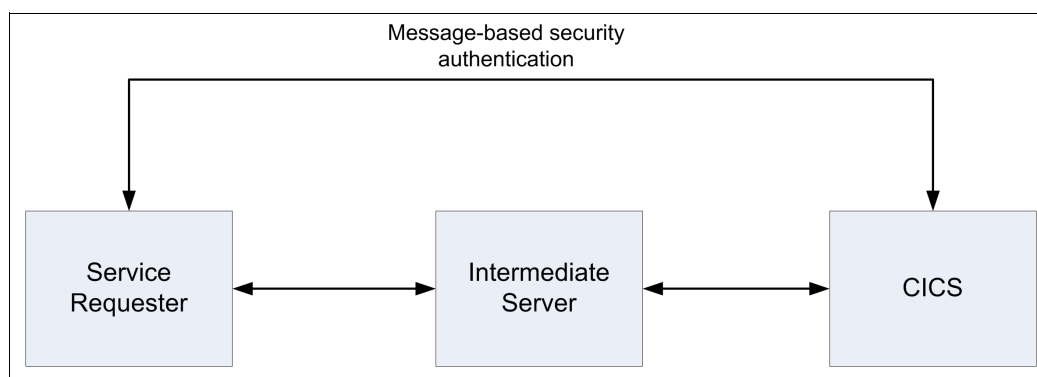


Figure 4-3 SOAP message-based security authentication

CICS supports the WS-Security¹ specification by adding new elements in SOAP header for protecting messages. WS-Security can be used to authenticate the service requester, to protect the integrity of XML data, and to ensure confidentiality of the transmitted message.

¹ The WS-Security specification is published by OASIS:
https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Figure 4-4 shows an example of how a SOAP message can be extended with security data that is used to authenticate the service requester and to protect the message as it passes between the service requester and the CICS service provider. The network portion of the diagram could contain any number of intermediate nodes, some of which might not be trusted.

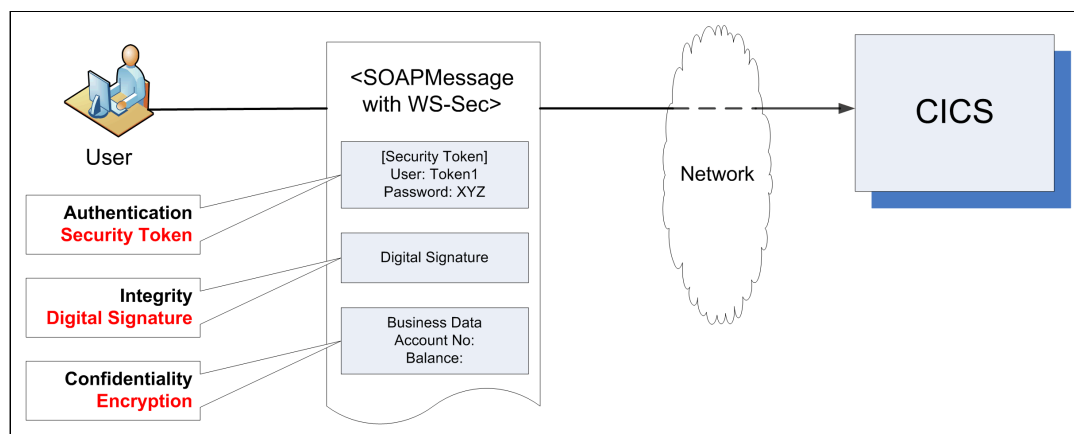


Figure 4-4 Example of SOAP message with WS-Security

CICS supports the following authentication options with message-based security:

- Basic authentication

CICS can accept a username token in the SOAP message header for authentication on inbound SOAP messages. The Username token contains a username element and a password element. CICS verifies the username and password using an external security manager such as RACF. If this is successful, CICS sets the user ID of the pipeline task to this value.

- Advanced authentication

CICS can exchange security tokens with a Security Token Service (STS) for authentication purposes. STS acts as a trusted third party, such as IBM Tivoli Federated Identity Manager, that provides standard based identity management services. The STS enables CICS to accept and send messages that have security tokens, such as Kerberos tokens, in the message header.

- X.509 certificates

An X.509 certificate that is used for signing can also be used for authentication. This type of security token is known as a *binary security token*. CICS maps the certificate to a RACF user ID and sets the user ID of the invoked transaction to this value.

- SAML assertion

The CICS TS feature pack for Security Token Extensions provides native support to validate SAML assertions and extract information from them. This feature is available for CICS TS V4.2 and later.

Identity assertion and propagation

Identity assertion is the method by which a distributed identity is mapped to a RACF identity and then asserted to CICS without a password check. Although identity assertion is a useful feature in a distributed environment involving intermediaries, the process of mapping distributed user identities to a RACF identity is typically a one-way function, resulting in the loss of the original distributed identity after the mapping occurs. Such mapping solutions have several issues, including lack of end-to-end accountability, inflexibility, and loss of control.

z/OS identity propagation is a newer form of identity assertion provided by z/OS 1.11 (V1R11) and later versions. It supports cross-platform, end-to-end security and provides for identity assertion, control, and auditing. Identity propagation addresses the issues associated with previous identity assertion solutions by allowing the z/OS security administrator to create a set of flexible rules, stored in the RACF database, to ensure that the distributed identity persists after the mapping stage and remains visible for operational support and auditing. We explain the auditing features of “Identity propagation” on page 110.

CICS JSON web services authentication

With the introduction on CICS TS Feature Pack for Mobile Extensions, CICS TS V4.2 and later provide JavaScript Object Notation (JSON) web services support for connecting mobile devices to CICS. JSON web services differ from traditional web services in that they use the JSON notation rather than XML and the SOAP protocol. JSON provides a lightweight and simple structure for messages rather than the heavyweight structure of XML, which makes it usable for mobile devices. For more information about CICS JSON web service, see *Implementing IBM CICS JSON Web Services for Mobile Applications*, SG24-8161.

The CICS JSON web services interface currently supports only HTTP transport protocol. Therefore, the interface uses the authentication mechanism provide by CICS web support for HTTP protocol, as described in “CICS web support authentication” in the next subsection.

CICS web support authentication

CICS web support is a collection of CICS services that enable a CICS region to communicate with clients over HTTP protocol. To configure CICS to act as a web server, the following resources are required to be defined for security authentication:

- ▶ TCPIPSERVICE
- ▶ URIMAP

TCPIPSERVICE

The TCPIPSERVICE definition can be configured to allow three types of security authentication by using AUTHENTICATE and SSL attributes:

- ▶ No authentication

This level of authentication is appropriate if the client is in a secure network and you can trust the client's local authentication mechanism. Specify AUTHENTICATE=NO setting for this configuration.

- ▶ Basic authentication

The client's identity is authenticated by the credentials provided by the client. This level of authentication is appropriate only if the credentials cannot be intercepted during transmission. The credential information is base-64 encoded. Specify AUTHENTICATE=BASIC for this configuration.

- ▶ SSL client certificate authentication

The client's identity is authenticated with a client certificate issued by a trusted third party (or certificate authority). Specify AUTHENTICATE=CLIENTAUTH and SSL=YES for this configuration.

URIMAP

A URIMAP definition is used by CICS web services to define the invoked transaction. The USERID attribute of URIMAP resource definition specifies the user ID under which the transaction runs.

CICS also provides the ability to write your own security analyzer program. This method can be used if the other methods are unsuitable. The analyzer program can be used to retrieve any authentication information from the HTTP request. It can also be used to determine what user ID should be attached to the invoked transaction.

IBM MQ authentication

IBM MQ (formerly known as WebSphere MQ) is a popular choice for integrating CICS applications with applications running in other servers, such as IBM WebSphere Application Server. The CICS MQ bridge (shown as the CICS-WebSphere MQ bridge in Figure 4-5) enables an application that is not running in a CICS environment to use WebSphere MQ messages to run a program or transaction on CICS and get a response. Figure 4-5 depicts the various steps that are required to process a single message and get the response.

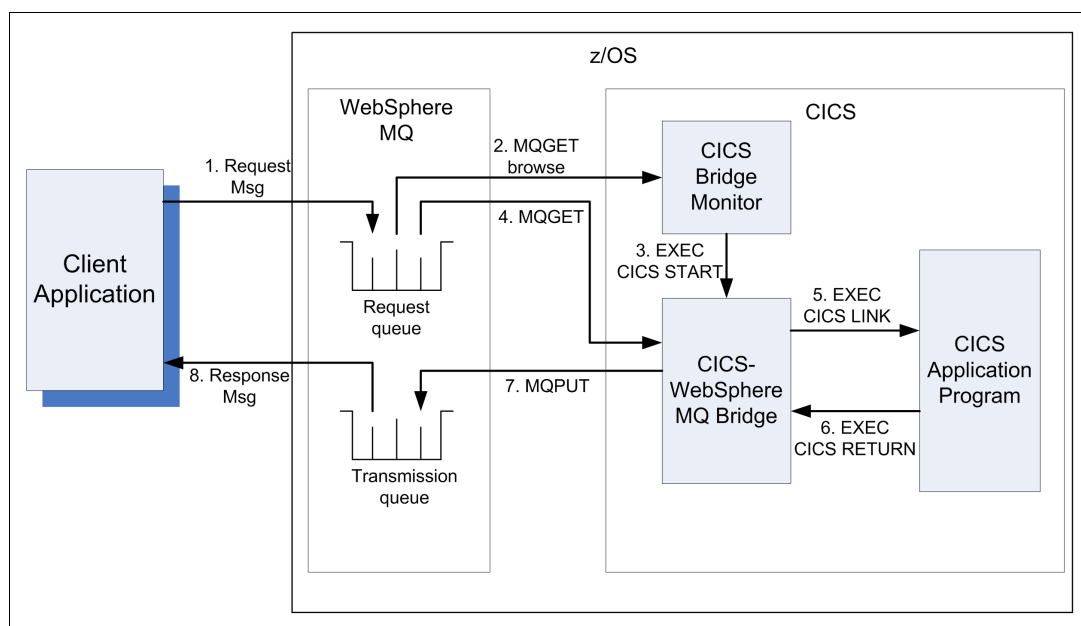


Figure 4-5 CICS IBM MQ bridge data flow

Let's examine the steps depicted in Figure 4-5:

1. A message, with a request to run a CICS program, is put on the request queue by the client application.
2. The bridge monitor task in CICS, which is constantly browsing the queue, recognizes that a "start unit of work" message is waiting (CorrelId=MQCI_NEW_SESSION).
3. Relevant authentication checks are made, and a CICS bridge task is started with the appropriate authority, which we describe later in this section.
4. The CICS bridge task reads and removes the message from the request queue.
5. The CICS bridge task issues an **EXEC CICS LINK** command for the program requested in the message and passes the request message to the task.
6. The CICS application program returns the response to the CICS bridge task.

7. The CICS bridge task uses the response to create a message and puts it on the transmission queue that corresponds to the reply-to queue specified in the request message. All reply messages (normal and error, requests and replies) are put in the reply-to queue with a default context.
8. The client application reads the response from the IBM MQ queue.

When you start the CICS MQ bridge, you can specify the following levels of authentication by using the AUTH parameter of the bridge monitor transaction:

► LOCAL

This level is the default. The bridge task and the CICS programs run by the bridge task are started with the CICS default user ID, so they run with the authority that is associated with this user ID. User IDs and passwords are not checked. If the bridge task runs a CICS program that tries to access protected resources, the CICS program might fail.

► IDENTIFY

The bridge monitor starts the bridge task with the user ID specified in the message descriptor (MQMD) in the request message. CICS programs that are run by the bridge run with the user ID from the MQMD. The password is not checked, and the user ID is treated as trusted.

► VERIFY_UOW

The bridge monitor checks the request message for the user ID in the MQMD and for the password in the IBM MQ CICS information header (MQCIH) before starting the bridge task with that user ID. CICS programs that are run by the bridge run with the user ID that is extracted from the MQMD. If the user ID or password is invalid, the request fails with return code MQCRC_SECURITY_ERROR. Only the first request message in the unit of work is checked; subsequent messages processed by the transaction are not checked.

► VERIFY_ALL

This level is the same as VERIFY_UOW, except that the bridge task checks the user ID and password in every message.

CICS TCP/IP sockets authentication

CICS TCP/IP sockets are normally used when the programmer needs close control over the TCP/IP communication between the client and the CICS application. CICS uses the z/OS Communications Server TCP/IP stack for socket communication.

Inherently, CICS does not provide any authentication mechanism for TCP/IP sockets. However, there are two options the programmer can use to ensure authentication of the remote client:

- Socket listener security exit
- Application Transparent Transport Layer Security (AT-TLS)

Socket listener security exit

The CICS socket interface provides a security exit that can be used to authenticate socket clients. The sockets listener transaction (CSKL) links to the security exit before starting the child server task. The security exit decides whether to allow the listener to start the server transaction based on the information it is passed, which includes this information:

Transaction ID	Transaction requested by client
Data area	User data received from client
Address	IP address of client
Socket	Socket descriptor

A user ID and password can be included in the data sent by the client. The security exit can then verify the user's credentials by using the **CICS VERIFY USER** command. The security exit can also be used to attach the child server task to this user ID.

Application Transparent Transport Layer Security (AT-TLS)

Because CICS uses z/OS Communications Server TCP/IP stack for socket communication, it can also use the authentication features provided by the z/OS Communications Server. The z/OS Communications Server TCP/IP stack provides Application Transparent Transport Layer Security (AT-TLS). This allows socket applications that use the TCP protocol to transparently use SSL/TLS to communicate with partners in the network.

Authorization

CICS provides a variety of mechanisms to secure transactions, resources such as files, and commands. It also provides methods to secure communication between clients and interfaces, such as IBM DB2 and IBM IMS.

CICS security checking can be controlled by using following CICS system initialization parameters:

- ▶ SEC
Set this parameter to YES to turn security check on for transactions and resources.
- ▶ SECPRFX
Set this parameter to YES if security profiles are defined in RACF with a prefix that corresponds to the user ID of the CICS region. You can also set the parameter to SECPRFX=*prefix* if the RACF security profiles are defined with any other prefix.

Transaction security

Transaction security is enabled by setting the XTRAN system initialization parameter. Specify XTRAN=YES or XTRAN=*resource_class_name* to control who can initiate transactions in CICS. If XTRAN=YES is specified, then CICS uses profiles defined in the RACF default classes, TCICSTRN and GCICSTRN. If you specify a resource class name, CICS uses the name that you specified, prefixed with a *T* for resource class and a *G* for grouping class.

When CICS transaction security is active, requests to attach transactions are associated with a user ID. When an Attach request is made, CICS calls RACF to determine whether the user belongs to the access list of the corresponding transaction profile in the RACF TCICSTRN class or group profile in the RACF GCICSTRN class.

The example shown in Figure 4-6 demonstrates RACF setup for transaction security. In this example, because XTRAN=YES is specified, the resource class name is TCICSTRN and the grouping class name is GCICSTRN. Transactions TRN1, TRN2, and TRN3 have profiles defined under class TCICSTRN. Transactions TRN4 and TRN5 are members of the GROUP1 group profile. Similarly, TRN6 and TRN7 are members of the GROUP2 group profile. GROUP1 and GROUP2 are group profiles under the GCICSTRN grouping class.

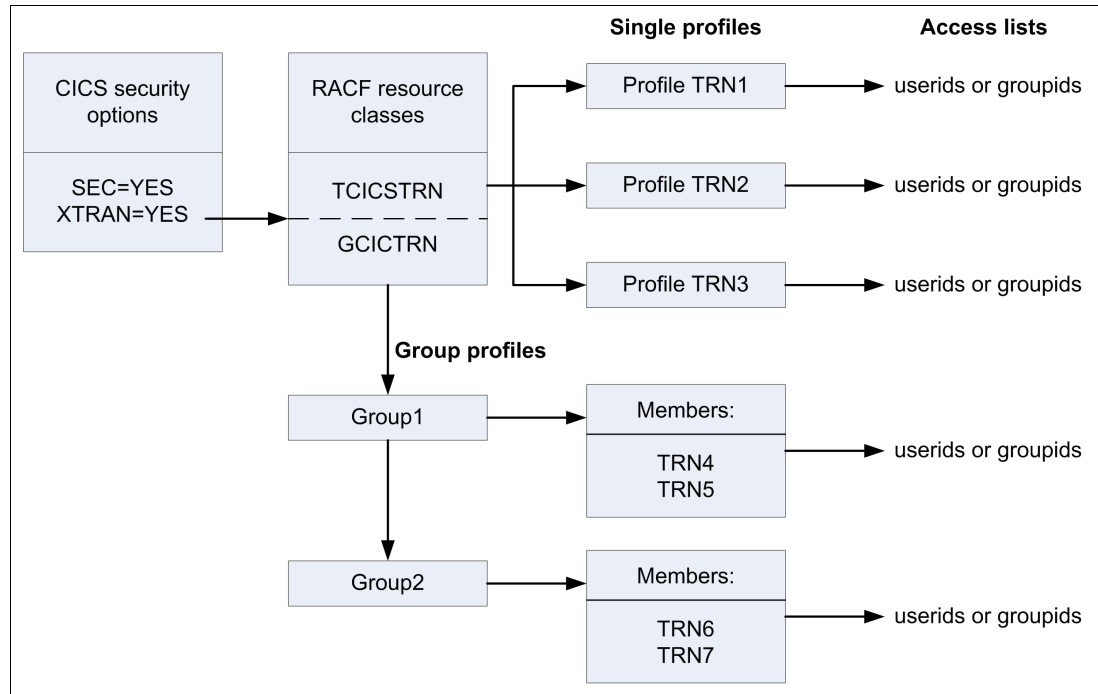


Figure 4-6 Transaction security RACF setup

Resource security

Resource security provides a further level of security to transaction security, by controlling access to the resources used by the CICS transactions. A user who is authorized to invoke a particular CICS transaction might not be authorized to access files, PSBs, or other general resources used within the transaction. Unlike transaction security, which cannot be turned off for individual transactions, you can control resource security checking at the individual transaction level.

Table 4-1 lists the resources that can be protected, along with the associated CICS system initialization parameter that required to be set for protecting the resource.

Table 4-1 System initialization parameters for resource security

Resource	CICS parameter	DFLT member class	DFLT group class
Partner logical units (LU6.2)	XAPPC	No defaults	No defaults
Application programming commands that are subject to command security	XCMD	CCICSCMD	VCICSCMD
DB2 resource class for DB2ENTRY	XDB2	No defaults	No defaults
CICS Transient data queue	XDCT	DCICSDCT	ECICSDCT
Enterprise bean methods	XEJB	No defaults	No defaults
CICS file-control-managed VSAM and BDAM files	XFCT	FCICSFCT	HCICSFCT
z/OS UNIX files managed by z/OS UNIX System Services	XHFS	No defaults	No defaults
CICS system log and general logs	XJCT	JCICSJCT	KCICSJCT
CICS started transactions and commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, INQUIRE TRANSACTION, INQUIRE REQID, SET TRANSACTION, and CANCEL	XPCT	ACICSPCT	BCICSPCT
CICS application programs	XPPT	MCICSPPT	NCICSPPT
DL/I program specification blocks (PSB)	XPSB	PCICSPSB	QCICSPSB
Resources that use XRES: ATOMOSERVICE, BUNDLE, DOCTEMPLATE, EPADAPTER, EVENTBINDING, JVMSERVER, and XMLTRANSFORM	XRES	RCICSRES	WCICSRES
CICS transactions	XTRAN	TCICSTRN	GCICSTRN
CICS temporary storage queues	XTST	SCICSTST	UCICSTST
Surrogate user security	XUSER	No defaults	No defaults

Command security

CICS command security applies to system programming commands, that is, commands that require the special CICS translator option, SP. Security checking is performed for these commands, when they are issued from a CICS application program, and for the equivalent commands that you can issue with the CEMT master terminal transaction.

Command security operates in addition to any transaction or resource security that you define for a transaction. For example, if a user is permitted to use a transaction called FILA, which issues an EXEC CICS INQUIRE FILE command that the user is not permitted to use, CICS issues a *not authorized* (NOTAUTH) condition in response to the command, and the command fails.

Surrogate user security

A surrogate user is one who has the authority to start work on behalf of another user. A surrogate user is authorized to act for that user without knowing that other user's password. To enable surrogate user checking, XUSER=YES must be specified as a system initialization parameter.

CICS performs surrogate user security checking in several situations, with the surrogate user facility of an external security manager (ESM), such as RACF. If surrogate user checking is in force, it applies to the following items:

- ▶ The CICS default user
- ▶ PLT post-initialization processing
- ▶ Preset terminal security
- ▶ Started transactions
- ▶ The user ID that is associated with a CICS business transaction services (BTS) process or activity that is started by a **RUN** command
- ▶ The user ID that is associated with a transient data destination
- ▶ The user ID that is supplied as a parameter on an **EXCI** call
- ▶ The user ID that is supplied in the AUTHID and COMAUTHID attributes of the DB2CONN and DB2ENTRY resource definitions
- ▶ The user ID that is supplied on the USERID attribute of URIMAP resource definitions

CICS DB2 interface

The CICS DB2 environment performs authorization security checks in various stages:

- ▶ Access to DB2 related resource in CICS
This could be a DB2CONN, DB2ENTRY, or DB2TRAN resource access by user, a transaction that accesses DB2 to obtain data, or a transaction that issues commands to the CICS DB2 attachment facility or to DB2. At this stage, CICS resource or command security is used to control access.
- ▶ Authorization IDs to DB2 for the CICS region or CICS transaction
Both the CICS region and the transaction must provide authorization IDs to DB2, and these authorization IDs are validated by RACF or an equivalent external security manager.
- ▶ Authorization for CICS users to access resources in DB2
The DB2 resource can be a plan, or a DB2 command, or the ability to execute dynamic SQL. At this stage, DB2 security checking is used to control the CICS user's access to the resource. DB2 security checking is done either by DB2 or by using an external security manager, such as RACF.

CICS IMS interface

CICS uses IMS Database Control (DBCTL) to connect to IMS databases. DBCTL is an IMS facility that provides an IMS Database Manager (IMS DM) subsystem that can be attached to CICS, but it runs in its own address spaces. DBCTL can be considered as an IMS Transaction Manager (TM) region without support for terminals.

When using CICS with DBCTL, the following optional security checks can be performed:

- PSB authorization check

At PSB scheduling time, CICS invokes security checking to determine whether the task user is authorized to access the PSB resource. This check is done at CICS before the request is seen by DBCTL, and it is similar to any other type of resource checking done in CICS. See Table 4-1 on page 104 for the RACF classes required for PSB resource authorization check.

- Resource access security checking by DBCTL

DBCTL resource access security checking provides the following checks:

- At connect time: Is the CICS user ID authorized to connect to IMS?
- At PSB schedule time: Is the CICS user ID authorized to access the PSB?

DBCTL can perform either or both of these checks by using either RACF or the IMS Resource Access Security exit routine (DFSRAS00).

Integrity and confidentiality

The z/OS architecture prevents individual regions, address spaces, from accessing data belonging to other regions. The integrity feature of z/OS prevents unauthorized access to protected storage with a region. CICS internally uses storage keys and storage protection to protect tasks from accessing other tasks storage.

Confidentiality ensures that an unauthorized party cannot obtain the meaning of the transferred or stored data. Typically, confidentiality is achieved by encrypting the data.

Storage protection

The z/OS subsystem storage protection facility helps you to prevent CICS code and control blocks from being overwritten accidentally by your application programs. CICS allows application programs to run in either user-key or CICS-key storage. CICS storage is automatically protected from being overwritten by application programs running in user-key storage (the default). The concept of isolating CICS code and control blocks (CICS internal data areas) from user application programs is illustrated in Figure 4-7.

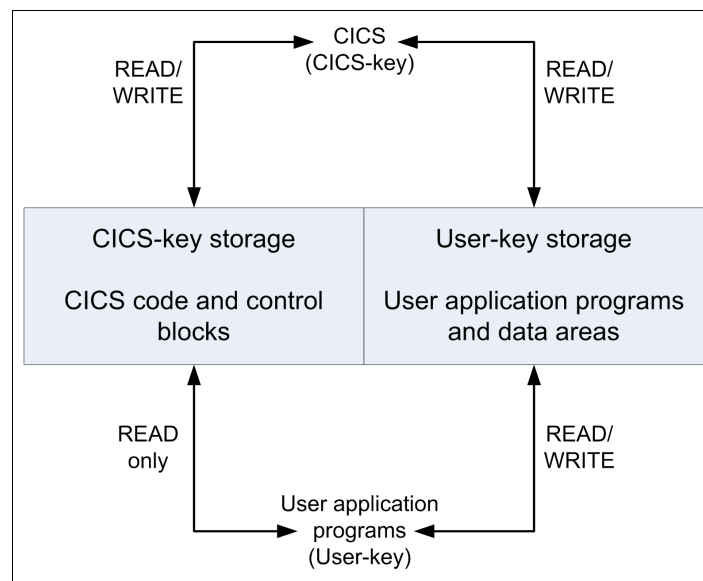


Figure 4-7 CICS storage protection

The use of storage protection is optional. You choose whether you want to use storage protection facilities by means of CICS system initialization parameters. The application program should be defined by using the proper EXECKEY attribute, and the transaction should be defined by using the proper TASKDATAKEY attribute.

Table 4-2 shows various combinations of EXECKEY and TASKDATAKEY that can be used.

Table 4-2 Key combinations

EXECKEY	TASKDATAKEY	Use
USER	USER	For normal user applications running under user-key.
USER	CICS	Not permitted. CICS abends any program defined with EXECKEY(USER) invoked under a transaction with TASKDATAKEY(CICS).
CICS	USER	For programs that need to issue restricted MVS requests or modify CICS-key storage.
CICS	CICS	For transactions and programs that function as extensions to CICS, or which require the same protection.

Transaction isolation

Transaction isolation extends this storage protection to provide protection for transaction data. Accidental overwriting of the transaction data by an application program of another transaction can affect the reliability and availability of your CICS system and the integrity of the data in the system. For transaction isolation, use the MVS subspace group facility.

A *subspace group* is a group of subspaces and a single base space. The subspace group facility uses hardware to provide protection for transaction data. The subspace group facility provides a partial mapping of the underlying base space so that only specified areas of storage in the base space are exposed in a particular subspace. Therefore, each subspace represents a different subset of the storage in the base space. When it is specified, transaction isolation ensures that programs defined with EXECKEY(USER) execute in their own subspace, with appropriate access to any shared storage or to CICS storage. A user transaction is limited to its own “view” of the address space.

In general, transaction isolation ensures that user-key programs are allocated to separate (unique) subspaces, and have the following levels of access:

- ▶ Read and write access to the user-key task-lifetime storage of their own tasks, which is allocated from one of the user dynamic storage areas (UDSA or EUDSA). They do *not* have any access to user-key task-lifetime storage of other tasks.
- ▶ Read and write access to shared storage; that is, storage obtained by **GETMAIN** commands with the **SHARED** option (SDSA or ESDSA).
- ▶ Read access to the CICS-key task-lifetime storage of other tasks (CDSA or ECDSA).
- ▶ Read access to CICS code.
- ▶ Read access to CICS control blocks that are accessible by the CICS API.

Encryption and cipher suites

When an SSL connection is established during the SSL handshake, the client and server exchange information about which cipher suites they have in common. They then communicate using the common cipher suite that offers the highest level of security. If they do not have a cipher suite in common, secure communication is not possible and CICS closes the connection.

The ENCRYPTION system initialization parameter is used to specify the level of encryption that CICS should use. The default value is STRONG, which means that CICS can use all two-character cipher suites to negotiate with clients. You can set a minimum and maximum encryption level by editing the list of cipher suites in the CIPHERS attribute on one of the following resource definitions:

- ▶ For inbound HTTP
Use the CIPHERS attribute of the TCPIP SERVICE resource definition, which automatically defines the PRIVACY attribute.
- ▶ For outbound HTTP and web service requests
Use the CIPHERS attribute of the URIMAP resource definition.
- ▶ For inbound IPIC
Use the CIPHERS attribute of the TCPIP SERVICE resource definition.
- ▶ For outbound IPIC
Use the CIPHERS attribute of the IPCONN resource definition.

The cipher suites that are supported by z/OS and CICS, and their characteristics are described in “Cipher Suite Definitions” in the IBM Knowledge Center:

<http://ibm.co/1KPRpno>

Auditing and monitoring

CICS auditing lets you capture and record security-related events (such as a user signing on or off of a system), so that they can be analyzed later, perhaps after a breach of security has occurred.

SMF logging

Except when processing certain security commands, CICS issues security authorization requests with the logging option. This means that RACF writes SMF type 80 log records to SMF. Which events are logged depends on the auditing in effect. For example, events requested by the AUDIT or GLOBALAUDIT operand in the resource profile or by the SETROPTS AUDIT or SETROPTS LOGOPTIONS command can be logged.

Security zSecure audit

As an automated security vulnerability analyzer for z/OS, IBM Security zSecure Audit for RACF provides the most comprehensive analysis available of a z/OS security posture by correlating data from various input sources.

Security zSecure Audit can correlate data from the following sources:

- ▶ Your external security manager (ESM) security databases
- ▶ The z/OS IPL parameters and other configuration information from multiple systems
- ▶ The System Management Facility (SMF) audit trail data from multiple systems
- ▶ Any sources (logs and flat files)

Security zSecure Audit is command-driven and uses the CARLa Auditing and Reporting Language (CARLa). The commands are explained in the *IBM Security zSecure Admin and Audit for RACF: User Reference Manual*, which is available in the IBM Security zSecure library in IBM Knowledge Center:

<http://ibm.co/10oNtzG>

A typical user who uses ISPF does not need to be concerned with CARLa. Security zSecure audit includes some standard reports for CICS. They can be accessed from Security zSecure menu *RE.3*. Figure 4-8 shows a selection window for CICS standard reports in Security zSecure audit.

Menu	Options	Info	Commands	Setup	Startpanel
Security zSecure Admin+Audit for RACF - Resource -					
CICS					
Option ===>					
R	Regions	CICS region reports			
T	Transactions	CICS transactions selection and reports			
P	Programs	CICS programs selection and reports			

Figure 4-8 Security zSecure audit CICS reports window

Figure 4-9 shows a sample CICS region report obtained by selecting option 1 from the previous window. Other CICS reports can be obtained by using the appropriate options.

```

COMMAND ==>                                SCROLL ==> CSR
***** Top of Data *****
CICS regions  26 Sep 2013 15:00                page    1
CICS region records for Jobnames, such as SC60CIC1, Systems like SC60

Region identification
-----
Complex name                PLEX60
System name                 SC60
CICS Region job name        SC60CIC1 Jobid STC17735 ASID 004E
CICS Region step name       SC60CIC1
VTAM Specific applid        SC60CIC1
VTAM Generic applid         SC60CIC1
VTAM CICSplex Generic applid
CICS System identification   WRKS
CICS System release level   TS 5.1.0
Default Userid              CICSUSER CICS      Dfltgrp: SYS1
Region Userid               IBMUSER           Dfltgrp: SYS1
PLT initialization userid    Dfltgrp:
  F1=HELP    F2=SPLIT  F3=END    F4=RETURN  F5=RFIND  F6=RCHANGE
  F7=UP      F8=DOWN   F9=SWAP   F10=LEFT  F11=RIGHT F12=RETRIEVE

```

Figure 4-9 Security zSecure audit CICS region report

Identity propagation

Identity propagation provides a mechanism to allow a user identity from an external security realm to be preserved, regardless of where the identity information was created, strengthening accountability across distributed environments.

In an external computing environment, for example WebSphere Application Server, the identity of a user is authenticated using a user identification that applies to that environment. Applications like WebSphere Application Server often use a separate, shared external security manager user ID when communicating with a CICS system. The original identity of the user is not passed to CICS and therefore cannot be passed onto the external security manager, which makes it difficult to determine the initial user identity and impacting the audit trail of the request.

The term *distributed identity* represents user identity information that originates from a remote system, for example, an X.500 distinguished name and associated LDAP realm. The distributed identity is created in one system and is passed to one or more other systems over a network. A distributed identity originates outside of CICS only; CICS is never the source of a distributed identity, but is capable of propagating the distributed identity onward.

When a distributed identity enters the sysplex over MRO and IPIC connections, it is automatically propagated in the sysplex, regardless of connection settings. CICS security handles the distributed identity as additional information relating to the user ID, and a distributed identity cannot exist without a user ID.

Figure 4-10 shows how the X.500 distinguished name and the associated LDAP realm, which identifies the user externally, are passed with the request from WebSphere Application Server to a CICS system. The distinguished name and realm, which are known in CICS as a distributed identity when they are transmitted across a network, are propagated into the z/OS security context and are associated with the RACF user ID. With the z/OS RACF command, **RACMAP**, you can use mapping filters to correlate the distinguished name and realm to a RACF user ID, preserve the distributed credential information, and fulfill governance and auditing requirements. RACF provides information to CICS about the distinguished name and realm, allowing retrieval in CICS of the identity of the initial user.

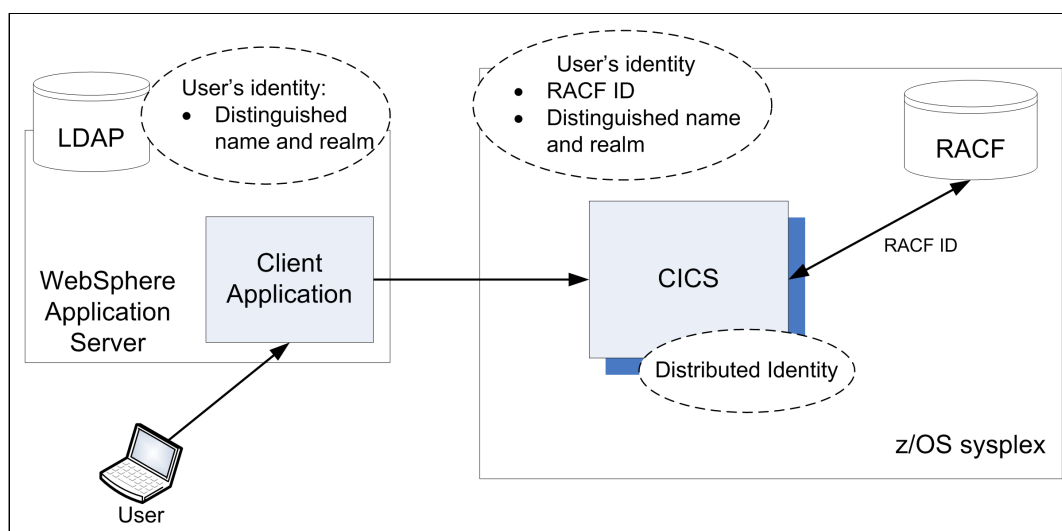


Figure 4-10 Identity propagation and distributed identity

Non-repudiation

Non-repudiation means that a sender and a receiver of data are able to provide legal proof to a third party that the sender did send the information and that the receiver received the identical information. Neither side is *able to deny*.

A solution for non-repudiation must provide proof of the integrity and origin of data and an authentication mechanism that with high assurance can be claimed to be genuine. The most common method of asserting the origin of data is through the use of digital signatures and a form of public key infrastructure (PKI).

4.1.2 Guiding principles for configuring security

In this section, we look at some of the common CICS mis-configurations and security considerations.

Common misconfigurations

A large number of security incidents in CICS originates because of improper configuration of security parameters, such as these examples:

- ▶ **Default user ID**

Ensure that the CICS default user ID specified in the SIT parameter DFLTUSER does not have high-level privileges. If a task is invoked without a user ID associated with it, CICS attaches DFLTUSER to the task.

Essentially, such tasks run without proper authentication. If the DFLTUSER has high-level privileges, the unauthenticated task can do many undesirable things. The solution to this problem is to ensure that the DFLTUSER has only bare-minimum privileges and authorization to CICS and z/OS resources.

- ▶ **Preset terminal security**

Preset terminal security should be used only for printers and devices without keyboards, because anybody who has physical access to the terminal can issue transactions that are authorized for the associated user ID on the terminal.

Security considerations

Next, let's consider when to use the various security features.

When to use authentication

Consider the following authentication-related questions:

- ▶ **Does the client need to authenticate?**

The answer can be decided for specific services rather than having a general rule for the application. It might be appropriate to run read-only services by using a generic user ID, but more sensitive services might need the requester to authenticate.

- ▶ **Who authenticates the service requester? CICS or an intermediary server?**

CICS can authenticate service requesters directly, or an intermediary might be able to provide an authentication service to CICS. In this case, the intermediary server authenticates the service requester and then transfers an *asserted* identity to CICS.

What authentication mechanism to use

For terminal-based applications and CICS intercommunication, there is no option other than RACF user ID and password authentication. For CICS web services and TCP/IP clients, you might choose to use transport-based security or message-based security or both. You might use only transport-based security to secure your environment in these circumstances:

- ▶ No intermediaries are used in the web service environment. Or, if there are intermediaries, you can guarantee that after the data is decrypted, it cannot be accessed by an untrusted node or process.
- ▶ The transport is based only on HTTP.
- ▶ Performance is your primary concern.
- ▶ The web services client is a stand-alone Java program.

WS-Security can be applied only to clients that run in a web services environment that supports the WS-Security specification (for example, WebSphere Application Server).

You might choose to use WS-Security (possibly in addition to transport-level security) in these circumstances:

- ▶ Intermediaries are used, some of which might be untrusted.
Security credentials that flow in the SOAP message can pass through any number of intermediaries. Protecting confidential information in the actual SOAP message can avoid the overhead of encrypting and decrypting through SSL at every intermediary node.
- ▶ Multiple transport protocols are used.
WS-Security works across multiple transports and is independent of the underlying transport protocol.
- ▶ You might choose to implement your own security procedures and processing by writing a custom message handler program that can process secure SOAP messages in the pipeline.
- ▶ If you chose SOAP message-based security, you may consider following types of tokens:
 - Username tokens, X.509 certificates, and ICRX (Extended Identity Context Reference) tokens, which can be processed directly by the CICS-supplied security handler.
 - You will probably need to configure the CICS-supplied handler to call an STS if other token types are used.

What security roles are required

In a CICS environment, the assets you normally want to protect are the application programs and the resources that are accessed by the application programs. You might want to provision authorities based on the roles of the user. The typical roles are system administrator, system operator, and client:

- ▶ System administrator
 - Needs authority to run CICS system programming commands and use CICS transactions used to perform administrative functions
 - Controlled by transaction and command security
- ▶ System operator
 - Needs authority to run CICS commands and CICS transactions used to operate and control the running CICS systems
 - Controlled by transaction and command security
- ▶ Client
 - Needs authority to run one or more client applications
 - Controlled by transaction and resource security

4.2 IBM Information Management System Transaction Manager

IBM Information Management System (IMS) consists of three components:

- ▶ IMS Database Manager (IMS DB)
- ▶ IMS Transaction Manager (IMS TM)
- ▶ IMS DB and IMS TM, a set of system services that provide common services

Often referred to collectively as *IMS DB/DC* (*DC* stems from the original name for the IMS Transaction Manager, Data Communications), these components comprise a complete online transaction and database processing environment that provides continuous availability and data integrity. IMS delivers accurate, consistent, timely, and critical information to application programs, which in turn deliver the information to many users and programs.

IMS TM and IMS DB can be ordered separately if both components are not required. The appropriate system services are provided for the components that are ordered. When IMS DB is ordered by itself, it is called DB Control (*DBCTL*). When IMS TM is ordered by itself, it is called *DC Control* (*DCCTL*).

IMS TM is a message-based transaction processor. It provides services to do the following tasks:

- ▶ Process input messages received from a variety of sources, such as the terminal network, other IMS systems, IBM MQ, and the web
- ▶ Process output messages that are created by application programs
- ▶ Provide an underlying queuing mechanism for handling these messages
- ▶ Provide interfaces to the TCP/IP network (IMS Connect)
- ▶ Provide high-volume, high-performance, high-capacity, low-cost transaction processing for both IMS DB hierarchical databases and DB2 relational databases

4.2.1 Security concepts and architecture

This section explains security concepts and features for IMS TM. For information about IMS DB security, see Chapter 2, “Database managers” on page 23. For IMS TM, two environment configurations are possible:

- ▶ DB/DC environment
- ▶ DCCTL environment

DB/DC environment

DB/DC environment includes both the IMS TM and IMS DB products working together. In the DB/DC environment, data is centrally managed for applications that are being executed concurrently and made available to terminal users.

A DB/DC environment consists of several regions:

- ▶ Control region

The IMS control region holds the control program, which continuously runs in the control region address space and controls the processing in other regions. Control region services DL/I database calls. Alternatively, DL/I can be run as a DL/I separate address space (DLISAS) region.

- ▶ DBRC region

Database Recovery Control (DBRC) facilities help manage database availability, data sharing, and system logging.

► Dependent regions

The dependent regions are separate address spaces from the control region, which are dependent on IMS and in which IMS schedules the applications that process transactions. Dependent regions are initiated by a z/OS **START** command or by a **/START REGION** command from the IMS master terminal. These are examples of dependent regions:

- Message processing programs(MPP) region
Online program processing region that can access full function databases
- Batch message processing (BMP)
Performs batch type processing online and access IMS message queues for input and output
- IMS fast path programs (IFP)
Similar to MPP but can quickly process and reply to messages from terminals
- Java message processing (JMP)
Similar to MPP except runs only Java or object-oriented COBOL programs
- Java batch processing (JBP)
Similar to BMP except runs only Java, object-oriented COBOL, or object-oriented PL/I programs

Also, DB/DC environment can have an optional address space for a coordinator controller (CCTL) and database resource adapter (DRA). CCTL is an interface that handles messages from external programs. CCTL uses the DRA to interface with the control region.

Figure 4-11 shows an example of a DB/DC environment.

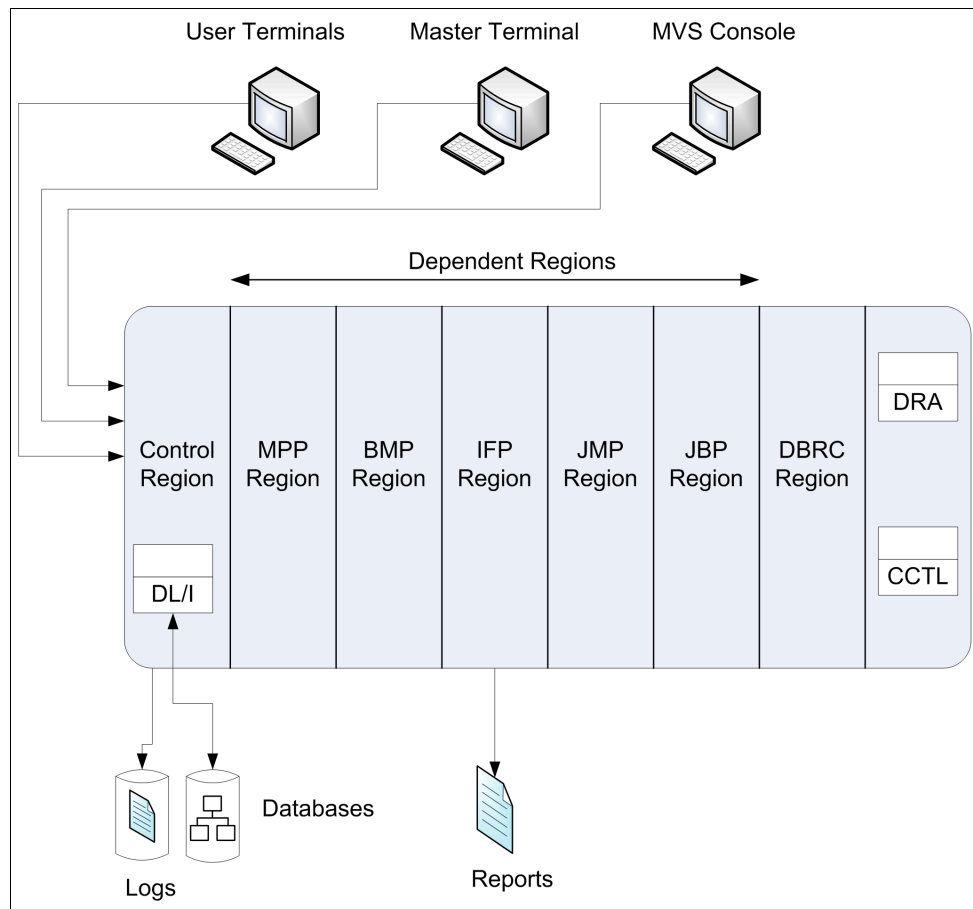


Figure 4-11 Example of a DB/DC environment

DCCTL environment

DCCTL is an IMS Transaction Manager subsystem that has no database components (IMS DB). A DCCTL environment is similar to the DB/DC environment. The primary difference is that a DCCTL control region owns no databases and does not service DL/I database calls.

Like DB/DC environment, DCCTL also consists of three types of address spaces:

- ▶ Control region
- ▶ DBRC
- ▶ Dependent regions

Figure 4-12 is an example of a DCCTL environment.

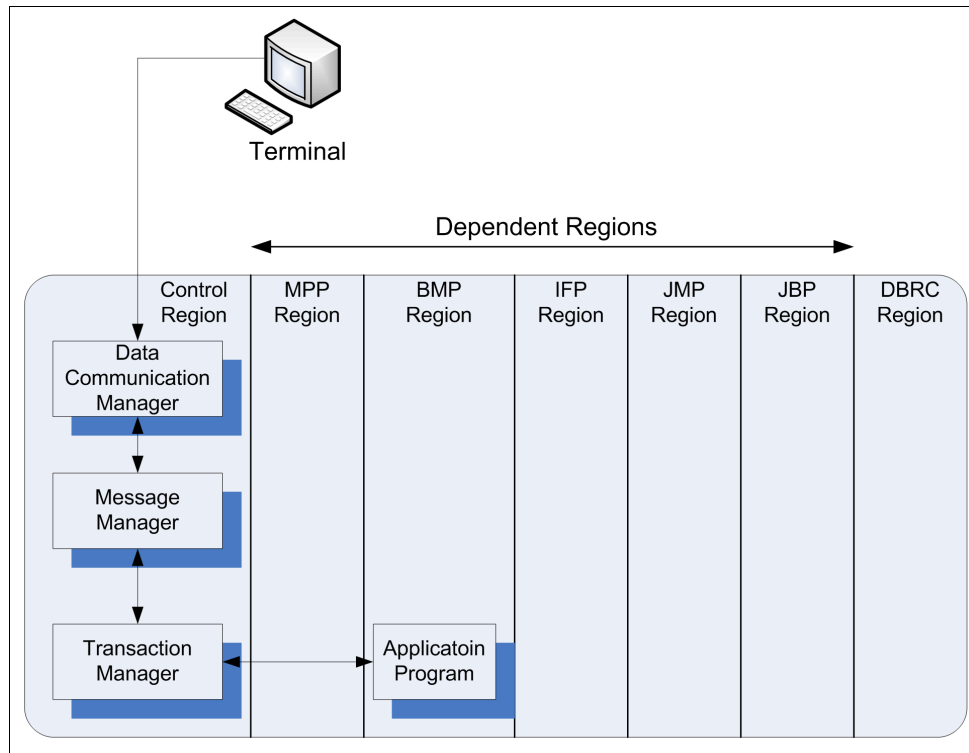


Figure 4-12 Example of a DCCTL environment

Implementing IMS security

You can configure the IMS system to use the RACF licensed program (or equivalent), an exit routine, or both to perform the following security functions:

- ▶ Securing IMS terminals
- ▶ Securing IMS commands
- ▶ Securing DBRC commands
- ▶ Securing PSBs and Online application programs

RACF security for IMS

A security plan for RACF includes defining which resources need to be protected, specifying security options in the IMS PROCLIB member data sets, and defining the resources to protect to RACF.

Use `RCLASS= keyword` to specify position 2 - 8 of the RACF resource classes that IMS will use. If `RCLASS= keyword` is omitted, the default value of IMS is used. For example, if you specify `RCLASS=ABC`, the command resource class is CABC, with the C prefixed to the RCLASS value of ABC.

If `RCLASS=` is not specified, the command resource class is CIMS. `RCLASS` can be specified in either the DFSPBxxx or DFSDCxxx member of the IMS PROCLIB data set. If specified in both places, the `RCLASS` value in DFSPBxxx is used.

Table 4-3 lists the resource class assignments, with the default RACF resource calls name and the user-defined name obtained from RCLASS= *keyword*.

Table 4-3 RACF resource class assignments

Resource class type	RACF-defined class	User-defined class (RCLASS= xxxxxxxx)
Command	CIMS	Cxxxxxxx
Command group	DIMS	Dxxxxxxx
LTERM	LIMS	Lxxxxxxx
LTERM group	MIMS	Mxxxxxxx
Transaction	TIMS	Txxxxxxx
Transaction group	GIMS	Gxxxxxxx
PSB	IIMS	Ixxxxxxx
PSB group	JIMS	Jxxxxxxx
Database	PIMS	Pxxxxxxx
Database group	QIMS	Qxxxxxxx
RESUME TPIPE call	RIMS	Rxxxxxxx
Other	OIMS	Oxxxxxxx
Other group	WIMS	Wxxxxxxx

Securing IMS terminals

To control access for a terminal control point, have the users identify themselves each time they use a terminal using signon verification. Signon verification security requires the entry of a user ID as a parameter on a /SIGN ON command at all terminals or a subset of the terminals.

To implement signon verification for statically defined VTAM terminals, set the OPTIONS= keyword to SIGNON in the TYPE or TERMINAL system definition macro. The specifications in the TERMINAL macro override those set in the TYPE macro. The default for OPTIONS= in the TYPE macro is NOSIGNON.

To avoid setting the OPTIONS= keyword in multiple system definition macros when it is required for all the static terminals to sign on, specify SIGNON=ALL in the DFSDCxxx PROCLIB member.

Signon verification is done by RACF, an exit routine (DFSCSGN0), or both. If you are using RACF, RACF checks for user ID, password, and group. If you are using an exit routine to implement sign-on verification, the exit routine can check for user ID and password.

With IMS Extended Terminal Support (ETO) terminals, the sign-on exit DFSSGNX0 can also be used. DFSSGNX0 is invoked after sign-on and before RACF, if RACF is also used for sign-on security checking. The exit could set the security bypass flag for the user, thereby bypassing sign-on security checking. For example, it is common for most installations to use the exit to bypass sign-on security for printers.

The PassTicket is an alternative to the RACF password and removes the need to send RACF passwords across the network in clear text. A RACF PassTicket is a one-time-only password that is generated by a requesting product or function. For more information about PassTicket, see “z/OS Security Server RACF Macros and Interfaces” in the IBM Knowledge Center:

<http://ibm.co/1L9DeuC>

Securing IMS commands

IMS supports two types of command formats to manage IMS systems and resources: IMS type-1 commands and IMS type-2 commands. IMS type-1 commands can be entered from IMS terminals, master terminals, system consoles, extended MCS consoles, IMS application programs through CMD calls and ICMD calls, and LU 6.2 and OTMA applications. IMS type-2 commands can be entered only from the Operations Manager (OM) address space, not from a master or remote terminal.

Type-1 commands are subject to security check by using the *CIMS* RACF class. The Command Authorization exit routine (DFSCCMD0) can also be used to provide a command authorization check. The Command Authorization exit routine can work in conjunction with RACF, or it can work independently, without RACF.

The Command Authorization exit routine is called for each IMS command. RACF is called first to perform the authorization. The return code is passed to the Command Authorization exit routine. DFSCCMD0 performs a final verification and determines the success or failure of the command authorization. DFSCCMD0 can also be used alone to perform the verification.

IMS provides the RVFY= parameter in the IMS procedure for customers who want to force reverification that the operator who signed on to a terminal is the same operator who is now entering a command or transaction. This reverification is done with RACF by including the word 'REVERIFY' in the APPLDATA field of the command profile as Example 4-1 shows.

Example 4-1 Reverification using RACF

```
RDEFINE Cxxx command UACC(NONE) APPLDATA('REVERIFY')
```

Both type-2 and type-1 commands can be entered through the OM address space. Security for commands entered through OM is controlled by the CMDSEC parameter specified in PROCLIB.

CMDSEC specified in the CSLOIxxx member tells OM whether to call RACF for all commands, both type-2 and type-1, using the RACF OPERCMDS class or OM security user exits. CMDSEC specified in the DFSCGxxx member tells IMS whether to call RACF for type-1 commands that come from OM using the CIMS class and the Command Authorization Exit if it exists. OM uses the OPERCMDS class for both type-1 and type-2 commands, as shown in Example 4-2.

Example 4-2

```
RDEFINE OPERCMDS IMS.plxname.command_verb.command_keyword UACC(NONE)
```

IMS uses the CIMS class for type-1 commands regardless of where they came from.

Securing IMS transactions

Transaction authorization determines whether a user ID is permitted to use a certain transaction. You can use an exit routine (DFSCTRNO), RACF, or both to perform the transaction authorization. IMS Version 13 removed the SECURITY macro and therefore removed the TRANEXIT keyword. There is no longer an explicit keyword to specify the use of DFSCTRNO. With IMS Version 13, if DFSCTRNO exists in RESLIB, it is called when the RACF return code is 0 (authorized) or 4 (undefined). If the RACF return code is 8 (not authorized), the Transaction Reverification Exit (DFSCTSE0) is called rather than DFSCTRNO.

Similar to the previous command security example, the REVERIFY option can be specified to RACF to force user to re-enter the signon password with each transaction code.

Securing DBRC commands

DBRC commands are a subset of IMS type-1 commands and can be secured at the command verb level in the RACF CIMS class in the same way all IMS type-1 transactions are secured. But IMS also provides specific DBRC command security that allows you to secure these commands at a more granular level whether you enter them online or with the DBRC utility (DSPURX00).

You record the security that you want in the RECON data set by using the CMDAUTH keyword of the CHANGE or INIT RECON. The setting of CMDAUTH tells DBRC whether to call RACF or the DBRC Command Authorization Exit (DSPDCAX0) or both. DBRC commands can be defined to RACF in the FACILITY class as Example 4-3 shows.

Example 4-3 Defining a DBRC command in the FACILITY class

```
RDEFINE FACILITY hlq.verb.modifier.qualifier UACC(NONE)
```

For a complete list, see “Resource names for command authorization” in the IBM Knowledge Center:

<http://ibm.co/1FJzgfn>

Securing PSBs and Online application programs

Resource Access Security (RAS) is used to authorize dependent regions to the resources they access including the IMS control region itself. Dependent regions include BMP, MPP, CICS, DB2 stored procedure, and so on.

You can use an exit routine (DFSRAS00), RACF, or both. When you activate RAS by setting the IMS start parameter ISIS=R or A, IMS calls RACF to check that the dependent region user ID is authorized to access the IMS ID when the dependent region requests to connect to IMS. The IMS ID can be protected in the RACF APPL class. This connection check is not done unless RAS security is activated.

If the dependent region connects successfully, RAS security can also be used to check the dependent region's user ID for authorization to the PSB, transaction or LTERM that it is accessing.

Access from other environments

External applications can access IMS through one of the following methods:

- ▶ APPC
- ▶ MSC and shared-queues
- ▶ OTMA
- ▶ IMS Connect

APPC

The IMS/APPC interface can be secured at several levels:

- ▶ VTAM APPC security options, which may be used to secure partner LUs in LU-LU sessions.
- ▶ MVS APPC security options which include conversation-level security and transaction program (TP) security options.
- ▶ IMS APPC security options for commands and transactions.

This section deals only with the IMS APPC security options. The APPC security level is set for all APPC/IMS activity using the **/SECURE APPC** command or the APPCSE= start parameter.

Unless overridden by the **/SECURE APPC** command or an APPCSE= start parameter, the default APPC security level is FULL. Commands entered from APPC are checked against the RACF CIMS and DIMS resource classes and are also candidates for evaluation by DFSCCMD0.

Transactions entered from APPC are checked against the RACF TIMS and GIMS resource classes and are candidates for evaluation by DFSCTRN0.

MSC and shared-queues

In Multiple Systems Coupling (MSC) and shared-queues environments that are made up of multiple IMS systems, each IMS system can receive transactions from terminals and from the other IMS systems in the same environment. The transactions from the other IMS systems require you to take additional security measures.

The MSCSEC= start parameter, which is found in the DFSDCxxx PROCLIB member, includes two positional parameters. The first positional parameter allows you to specify (based on the MSC transaction type) whether and when a receiving IMS calls RACF and exit routines to check security for a transaction.

There are two types of transactions that can be received on an MSC link for which you might check security:

- ▶ Direct-routed
- ▶ Non-direct-routed

The first positional parameter of MSCSEC= specifies the type of transaction to protect. These are the options:

- ▶ LRDIRECT

RACF and exit routines check security on non-direct-routed transactions and no security checking is performed for direct-routed transactions. This is the default.

- ▶ LRNONDR

RACF and exit routines check security on direct-routed transactions and no security checking is performed for non-direct-routed transactions.

- ▶ LRALL

RACF and exit routines check both types of transactions.

- ▶ LRNONE

The IMS system does not request any security checking for either type of transaction.

The second positional parameter of the MSCSEC= start parameter specifies the user ID on which to base security checking. These are the options:

- ▶ CTL

This specifies that RACF and exit routines use the user ID of the receiving IMS control region. This is the default.

- ▶ MSN

This specifies that RACF and exit routines use the MSNAME in the received message as the user ID.

- ▶ USR

This specifies that RACF and exit routines use the ID of the user who originally initiated the transaction at the inputting terminal.

The TM and MSC Message Routing and Control exit routine (DFSMSCE0) is concerned primarily with the routing of messages for TM and MSC. However, you can also use this exit to specify which user ID to use for security checking. The DFSMSCE0 specifications for user IDs are the same as those specified by the second-position parameter of the MSCSEC= keyword in the DFSDCxxx PROCLIB member.

OTMA

Open Transaction Manager Access (OTMA) is a function of IMS that was introduced with IMS Version 5. OTMA is a transaction-based, connectionless client/server protocol that provides an access path and an interface specification for sending and receiving transactions and data from IMS.

OTMA is specifically implemented for IMS in an z/OS sysplex-capable environment. Therefore, the domain of OTMA is restricted to the domain of the z/OS cross-system coupling facility (XCF). XCF is a component of z/OS that provides functions to support cooperation between authorized programs running within a sysplex. OTMA and z/OS applications compose an XCF group, where OTMA and the applications are group members.

In a Parallel Sysplex environment, different members can be on different z/OS images. Figure 4-13 shows examples of OTMA client applications and corresponding network client applications.

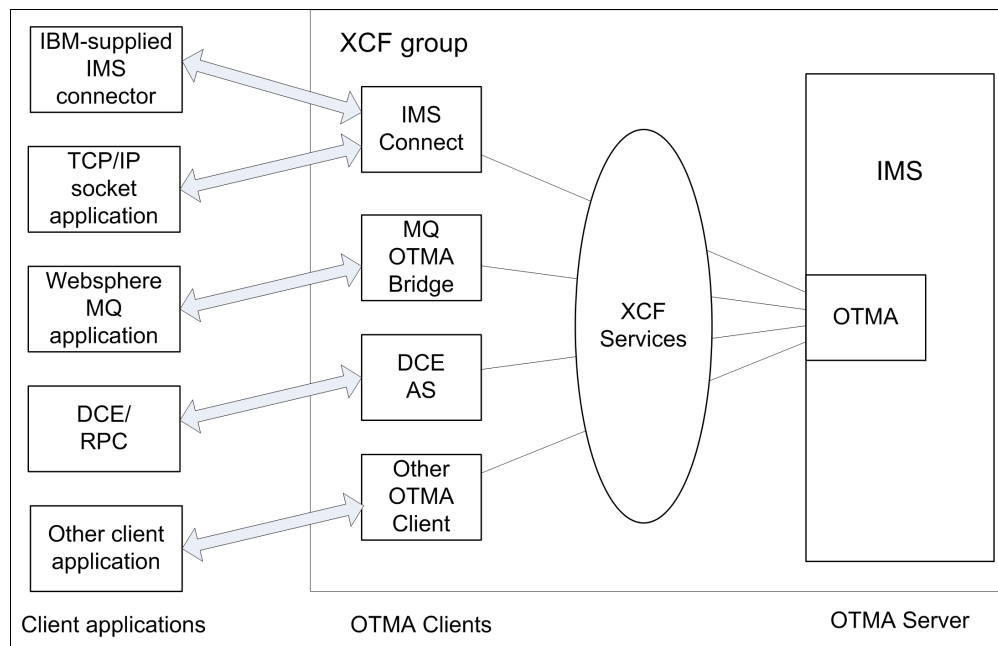


Figure 4-13 Example IMS OTMA configuration

When a client sends a client bid command to OTMA to connect, OTMA has to verify that the client is sufficiently authorized to connect to OTMA. The data passed in the security-data section of the client bid message is used by RACF to verify the client's authorization. The `IMSXCF.xcf_roup_name.xcf_member_name (client)` must be defined to RACF in the FACILITY class. When OTMA processes an input message, whether it is command or transaction input, security is driven by these factors:

- ▶ Status as set by the OTMASE= parameter in DFSPBxxx member
- ▶ Status as set by the IMS /SECURE OTMA command
- ▶ Security data in the OTMA header

These are the OTMASE parameters:

- ▶ CHECK

Causes existing RACF calls to be made. IMS commands are checked by using the RACF resource class of CIMS. IMS transactions are checked by using TIMS.
- ▶ FULL

Causes the same processing as the CHECK parameter but uses additional RACF calls to create the security environment for dependent regions.
- ▶ NONE

Does not call RACF within IMS for security verification.
- ▶ PROFILE

Causes the values in the security-data section of the OTMA message prefix for each transaction to be used.

The `/DISPLAY OTMA` command shows the security level that is currently in effect.

IMS Connect

IMS Connect is a feature of IMS that provides high-performance communications for IMS between one or more TCP/IP or local z/OS clients and one or more IMS systems. IMS Connect has the following features:

- ▶ Provides commands to manage the communication environment
- ▶ Assists with workload balancing
- ▶ Supports multiple TCP/IP clients accessing multiple datastore resources
- ▶ Reduces design and coding efforts for client applications
- ▶ Offers easier access to IMS applications and operations with advanced security and transactional integrity

IMS Connect uses OTMA for accessing IMS. The architecture supports any TCP/IP clients that are communicating with socket calls. IMS Connect also supports the TCP/IP clients that are using the IMS Connector for Java.

Before IMS Connect can send any messages to IMS through OTMA for processing, IMS Connect has to identify itself as an OTMA client. This is achieved by issuing an **OTMA** command of the type *client bid* and passing the required security data to OTMA for verification. The user ID passed by IMS Connect in the client bid must have READ access to the FACILITIES class entry `IMSXCF.xcf_group_name.xcf_member_name (client)` in RACF.

After IMS Connect has successfully joined the XCF group and connected as an OTMA client to IMS, there is the option to let IMS Connect do RACF user ID and password verification of each client on a per-message basis. This facility is driven by the `RACF=Y | N` parameter, as specified in the `HWSCFG` configuration file. You can modify the RACF status by usJavaing the IMS Connect command **SETRACF=ON | OFF**.

Event monitoring and recording

The system log data sets are a basic source for statistics about the processing performed by the online system. IMS logs specific records for sign-on and security violations. IMS logs record type x'16' for signon and sign-off activity and logs record type x'10' for security violation. Individual log record types contain data that can be analyzed in many ways. For example, you can select and format all activity pertaining to a user ID.

IMS DB/DC and DCCTL environment provides several utilities to assist with extracting log records:

- ▶ Log Transaction Analysis utility
Log Transaction Analysis utility (DFSILTA0) collects information about individual transactions, based on records in the system log. Many events are tabulated in the Log Analysis report that is produced by this utility, including total response time, time on the input queue, processing time, and time on the output queue.
- ▶ Statistical Analysis utility
Statistical Analysis utility (DFSISTS0) produces several summary reports. These reports contain actual transaction loads and response times for the system. The statistics produced are dependent on the input system log data sets.
- ▶ IMS Monitor
The IMS Monitor collects data while the online IMS subsystem is running. It gathers information for all dispatch events and places it, in the form of IMS Monitor records, in a sequential data set. Reports based on the IMS Monitor output can be obtained using the IMS Performance Analyzer for z/OS or the IMS Monitor Report Print utility (DFSUTR20).

4.2.2 Guiding principles for configuring security

In this section, we explain common IMS misconfigurations and security considerations.

Common misconfigurations

The following examples are among the common IMS security misconfigurations:

- RACF resource class not activated

Even if you have not defined any profiles in a required resource class, the resource class must have been activated in RACF. If it hasn't been, IMS fails to initialize (U0166).

Example 4-4 Activating the resource class in RACF

```
SETR CLASSACT(CIMS)
```

- RACF profiles not defined

IMS allows access to a resource if the corresponding RACF profile is not defined. RACF sends a return code of 4 when a resource is not defined to RACF. IMS treats return code 4 the same as return code 0, so it allows access to the resource.

DBRC does not allow access to a resource if the corresponding RACF profile is not defined. DBRC treats return code 4 the same as return code 8 (not authorized).

Be sure to define profiles in RACF before you activate DBRC command security.

- Refresh RACF data space

RACF updates are not visible until a RACF refresh is done. Recycling IMS region does not refresh RACF resource profiles defined for IMS. To refresh the RACF data space, issue the commands as shown in Example 4-5. The example shows the commands to refresh the CIMS class in RACF. Unless RACF is configured for sysplex communication, the refresh must be done on all LPARs that need to see the update.

Example 4-5 RACF refresh command

```
SETR RACLIST CLASS(CIMS) REFRESH  
SETR GENERIC CIMS REFRESH
```

- Invoking exits

If a security parameter does not have a setting to explicitly specify an exit, the exit will be invoked if it exists in RESLIB.

If a security parameter is set to explicitly specify an exit, the exit must exist or IMS will fail to initialize (U0718).

Security considerations

This section explains how and when to use the security features described in this chapter.

Security considerations for the master terminal

The security of access from the master terminal is *critical*. Because the master terminal operator (MTO) can modify all security profiles during normal operations, consider protecting the terminal with a second level of control. Signon verification security provides this capability.

The primary question is how much capability to modify security should be given to this second level of control. Default security does not and cannot prevent modifying the system's security profiles through the master terminal; however, you might want to restrict some commands from being entered from the MTO. You can use the DFSCCMD0 exit routine to limit the commands that can be entered from the MTO. To control transactions being issued from MTO, assign a user ID to the MTO by using the MTOUSID parameter in DFSDCxxx. This allows you to use RACF to authorize the MTO to transactions.



IBM MQ messaging system

In this chapter, we review security concepts and provide an architectural overview for IBM MQ messaging system on z Systems. Then, we describe guiding principles for configuring messaging system security.

This chapter includes the following main sections:

- ▶ IBM MQ security concepts and architecture
- ▶ Guiding principles for configuring security

5.1 IBM MQ security concepts and architecture

IBM MQ for z/OS uses the z/OS System Authorization Facility (SAF) to provide access control services within the z/OS environment. IBM MQ does no security verification of its own. Instead, it uses an external security manager (ESM), such as z/OS RACF Security Server, through the SAF interface.

5.1.1 Security setup

The default security setup in IBM MQ is that all users can access and change every resource. This includes not only the local users but also those on remote systems that are using distributed queuing or clients, where the logon security controls might be less strict than is normally the case for z/OS.

In this section, we explain how to enable security for IBM MQ by using RACF. We describe the following RACF resources that are required for IBM MQ security and how to configure them:

- ▶ “IBM MQ RACF classes”
- ▶ “IBM MQ RACF profiles”
- ▶ “IBM MQ RACF switch profiles” on page 129
- ▶ “IBM MQ RACF RESLEVEL profile” on page 133

IBM MQ RACF classes

RACF classes are used to hold profiles required for WebSphere MQ security checking. Each RACF class holds one or more profiles used at some point in the checking sequence.

Table 5-1 shows the WebSphere MQ RACF classes and their purposes.

Table 5-1 WebSphere MQ RACF classes

Member class	Group class	Purpose
MQADMIN	GMQADMIN	Profiles used for administrative functions
MQCONN		Profiles used for connection security
MQCMDS		Profiles used for command security
MQQUEUE	GMQQUEUE	Profiles used for queue resource security
MQPROC	GMQPROC	Profiles used for process resource security
MQNLIST	GMQNLIST	Profiles used for namelist resource security

Some classes have a related *group class* that is used for grouping resources that have similar access requirements. For more information, see the RACF security administrator's guide.

The classes must be activated before the security check can be made. To activate all of the IBM MQ classes, use the following command:

```
SETROPTS CLASSACT(MQADMIN,MQCONN,MQCMDS,MQQUEUE,MQPROC, MQNLIST)
```

IBM MQ RACF profiles

The IBM MQ RACF profiles are used to control access to various IBM MQ resources. The IBM MQ resources that need to be protected should have a corresponding IBM MQ RACF profile under the appropriate IBM MQ RACF class.

Security in IBM MQ can be implemented at the queue manager level or at the queue-sharing group level. If security is implemented at the queue-sharing level, all the queue managers in the group share the same RACF profile. At queue-sharing level, it is also possible to override the default queue-sharing level security settings for individual queue managers of the group.

All profiles used by IBM MQ contain a prefix. The prefix is the queue manager name for queue manager level security. For queue-sharing group level security, the prefix is the queue-sharing group name.

For example. If you want to protect a queue called *QUEUE_A* in the QSG1 queue-sharing group at the queue-sharing group level, the appropriate profile would be defined this way in RACF:

```
RDEFINE MQQUEUE QSG1.QUEUE_A
```

If *QUEUE_A* belongs to queue manager QM01 at queue manager level security, the RACF profile would be defined this way:

```
RDEFINE MQQUEUE QM01.QUEUE_A
```

In RACF you can define generic profiles for protecting a large number of resources. For example, you can define a single RACF profile, *QM01.QUEUE_**, to protect all queues starting with the name *QUEUE_* under queue manager QM01.

IBM MQ RACF switch profiles

A switch profile is a normal RACF profile that has a special meaning to IBM MQ. Unlike regular profiles, the access list in switch profile is not used by IBM MQ. Instead, switch profiles are used as indicators by IBM MQ to determine whether certain security features should be turned ON or OFF.

Switch profiles are defined under the MQADMIN class. When a queue manager is started, it checks the MQADMIN class to see whether any switch profiles have been defined. The profiles determine whether the corresponding IBM MQ security is set off and that type of security is deactivated. If any switch is set on, IBM MQ checks the status of the RACF class associated with the type of security corresponding to the switch.

To set security switch off, you need to define a *NO.** switch profile for it. The existence of a *NO.** profile means that security checks are not performed for that type of resource. For example, process security is deactivated if the *NO.PROCESS.CHECKS* switch profile is defined. At queue-sharing level security, you can override the security setting for a particular queue manager by defining the *qmgr-name.YES.** profile.

The first security check made by IBM MQ is used to determine whether security checks are required for the whole IBM MQ subsystem. If you specify that you do not want subsystem security, no further checks are made. Table 5-2 lists the various switch profiles for subsystem security.

Table 5-2 Switch profiles for subsystem security

Switch profile name	Type of resource that is controlled
qmgr-name.NO.SUBSYS.SECURITY	Subsystem security at queue manager level deactivated
qsg-name.NO.SUBSYS.SECURITY	Subsystem security at queue-sharing level deactivated
qmgr-name.YES.SUBSYS.SECURITY	Subsystem security overridden for the queue manager at queue-sharing level

Figure 5-1 shows the order in which subsystem security is checked.

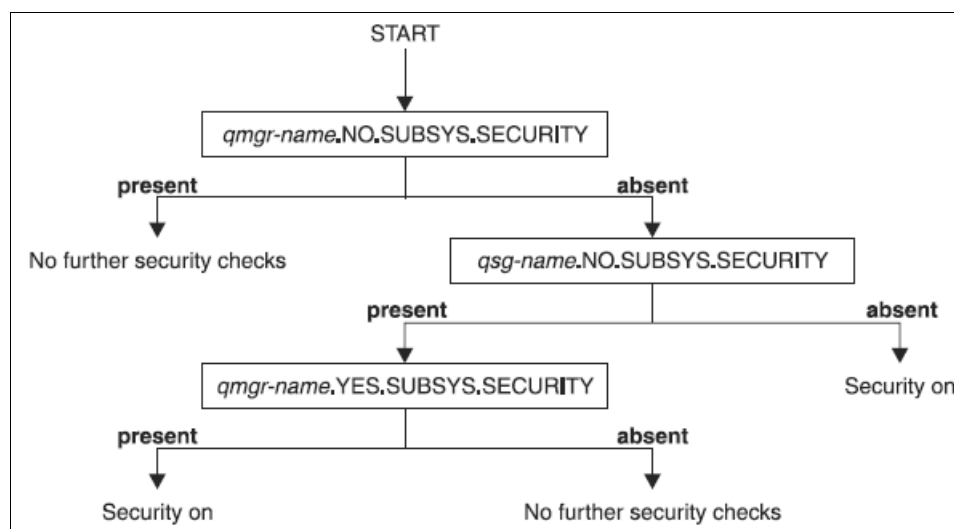


Figure 5-1 Checking for subsystem security

When IBM MQ has determined that security checking is required, it then determines whether checking is required at queue-sharing group or queue manager level, or both. These checks are not performed if your queue manager is not a member of a queue sharing group.

Table 5-3 lists the switch profiles that determine the level of security required.

Table 5-3 Switch profiles for queue-sharing group and queue manager level security

Switch profile name	Type of checking that is controlled
qmgr-name.NO.QMGR.CHECKS	No queue manager level checks for this queue manager
qsg-name.NO.QMGR.CHECKS	No queue manager level checks for this queue-sharing group
qmgr-name.YES.QMGR.CHECKS	Queue manager level checks active for this queue manager
qmgr-name.NO.QSG.CHECKS	No queue-sharing group level checks for this queue manager
qsg-name.NO.QSG.CHECKS	No queue-sharing group level checks for this queue-sharing group
qmgr-name.YES.QSG.CHECKS	Queue-sharing group level checks overridden for this queue manager

Figure 5-2 shows the order in which queue manager level security is checked.

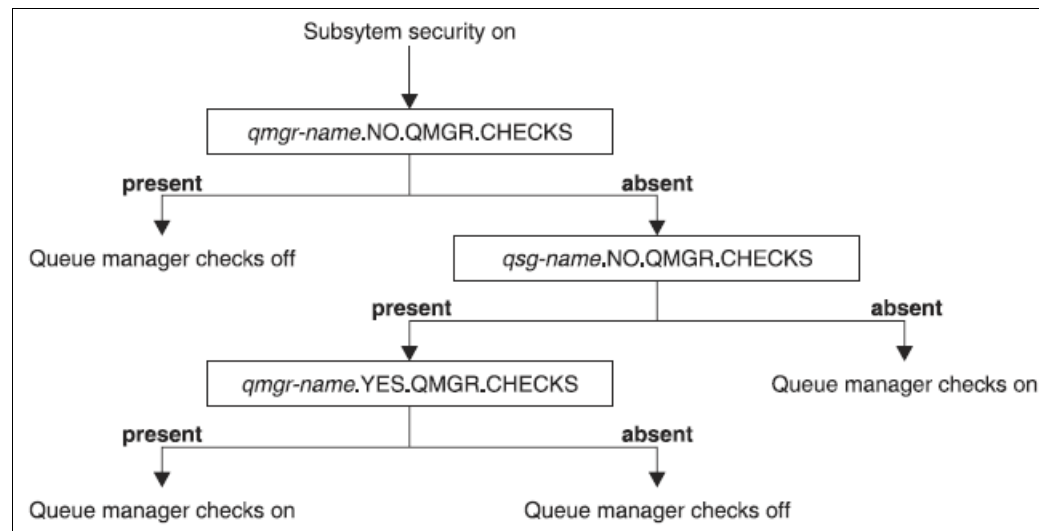


Figure 5-2 Checking for queue manager level security

Figure 5-3 shows the order in which queue-sharing group level security is checked.

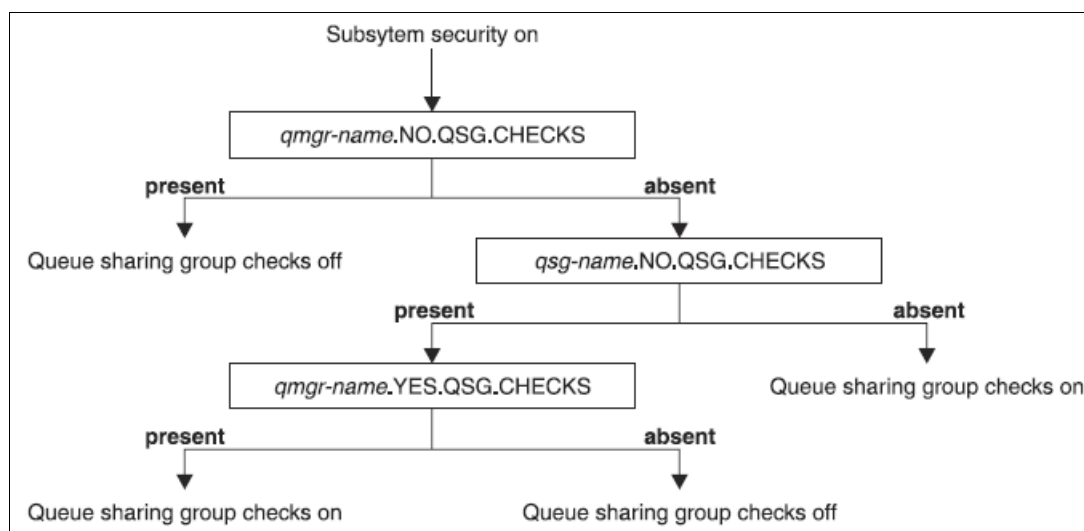


Figure 5-3 Checking for queue-sharing group level security

After IBM MQ has determined that subsystem security is active and the level of security checking required for the queue manager or queue-sharing group, IBM MQ looks for switch profiles for resource security checks.

Table 5-4 shows the switch profiles that are used to control access to IBM MQ resources. Some profiles apply to both queue manager and queue-sharing groups. These are prefixed with *hlq*, so substitute the queue manager name or the queue-sharing group name for *name*. Profiles with a *qmgr-name* prefix are the queue manager overrides, where you specify the queue manager name.

Table 5-4 Switch profiles for resource checking

IBM MQ resource type	Switch profile name	Switch profile for queue manager override
Connection security	hlq.NO.CONNECT.CHECKS	qmgr-name.YES.CONNECT.CHECKS
Queue security	hlq.NO.QUEUE.CHECKS	qmgr-name.YES.QUEUE.CHECKS
Process security	hlq.NO.PROCESS.CHECKS	qmgr-name.YES.PROCESS.CHECKS
Namelist security	hlq.NO.NLIST.CHECKS	qmgr-name.YES.NLIST.CHECKS
Context security	hlq.NO.CONTEXT.CHECKS	qmgr-name.YES.CONTEXT.CHECKS
Alternate user security	hlq.NO.ALTERNATE.USER.CHECKS	qmgr-name.YES.ALTERNATE.USER.CHECKS
Command security	hlq.NO.CMD.CHECKS	qmgr-name.YES.CMD.CHECKS
Command resource security	hlq.NO.CMD.RESC.CHECKS	qmgr-name.YES.CMD.RESC.CHECKS

5.1.2 IBM MQ RACF RESLEVEL profile

RESLEVEL is a RACF profile that controls the number of user IDs checked for IBM MQ resource security. Normally, when a user attempts to access an IBM MQ resource, RACF checks the relevant user ID or IDs to see if access is allowed to that resource. By defining a RESLEVEL profile you can control whether zero, one or, where applicable, two user IDs are checked for IBM MQ resource access.

When an application tries to connect to IBM MQ, IBM MQ checks the access that the user ID associated with the connection has to a profile in the MQADMIN class called *hlq.RESLEVEL*, where *hlq* is either the queue manager name or the queue-sharing group name.

If there is a RESLEVEL profile, the level of checking depends on the connection environment and the type of access to the profile. To activate checking of user IDs for resource access, you should define a RESLEVEL (either at the queue manager level or the queue-sharing group level) with a UACC(NONE) and ensure that the relevant user ID or IDs have the proper access to the RESLEVEL profile.

RESLEVEL controls are checked on a connection-by-connection basis and last for the duration of the connection. The user IDs used for checking access to the RESLEVEL profile depend on the connection type, as described in the following list:

- Batch connection

The user ID is the ID running the batch job or task. Whenever the batch job or task tries to access an IBM MQ resource, resource checking is performed if the user ID of the job or task has access, as described in Table 5-5, to the RESLEVEL profile of the queue manager or queue-sharing group.

Table 5-5 RESLEVEL access levels for batch connection

RACF access level	Level of checking
NONE	Resource checks performed
READ	Resource checks performed
UPDATE	Resource checks performed
CONTROL	No checks
ALTER	No checks

- CICS connection

By default, when a resource security check is made on a CICS connection, two user IDs are checked to see if access is allowed to the resource.

The first user ID checked is that of the CICS address space. This is the user ID on the job card of the CICS job, or the user ID assigned to the CICS started task by the z/OS *STARTED* class or the started procedures table. The second user ID checked is the user ID associated with the CICS task or the alternate user ID.

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested.

For RESLEVEL profile access check, the CICS address space user ID is used. The level of access of the CICS address space user ID to the RESLEVEL profile determines whether two, one, or zero user IDs are checked for resource security. This is described in Table 5-6.

Table 5-6 RESLEVEL access levels for CICS connection

RACF access level	Level of checking
NONE	Check both the CICS address space user ID and the task or alternate user ID
READ	Check only the CICS address space user ID
UPDATE	Check the CICS address space user ID, and if the transaction is defined in CICS with RESSEC=YES, also check the task or alternate user ID
CONTROL	No checks
ALTER	No checks

► **IMS connection**

By default, two user IDs are checked for resource security when connection is made through IMS. The first user ID is that of the IMS region address space. The second user ID is associated with the IMS transaction, terminal, or the program specification block (PSB) name.

For RESLEVEL profile access check, the IMS address space user ID is used. The level of access of the IMS address space user ID to the RESLEVEL profile determines whether two, one, or zero user IDs are checked for resource security. This is described in Table 5-7.

Table 5-7 RESLEVEL access levels for IMS connection

RACF access level	Level of checking
NONE	Check both the IMS address space user ID and the transaction, terminal ID or PSB ID
READ	Check only the IMS address space user ID
UPDATE	Check only the IMS address space user ID
CONTROL	No checks
ALTER	No checks

► **Channel initiator connection**

By default, two user IDs are checked for resource security when connection is made through a channel initiator.

The first user ID is that of the channel initiator address space. The second user ID can be that specified by the MCAUSER channel attribute, that received from the network, or the alternate user ID for the message descriptor. This depends on the communication protocol you are using and the setting of the PUTAUT channel attribute.

For RESLEVEL profile access check, the channel initiator address space user ID is used. The level of access of the address space user ID to the RESLEVEL profile determines whether two, one, or zero user IDs are checked for resource security. This is described in Table 5-8.

Table 5-8 RESLEVEL access levels for channel initiator connection

RACF access level	Level of checking
NONE	Check both user IDs
READ	Check only one user ID based on the setting of PUTAUT channel attribute
UPDATE	Check only one user ID based on the setting of PUTAUT channel attribute
CONTROL	No checks
ALTER	No checks

5.1.3 IBM MQ resource security

You must define RACF profiles to control access to IBM MQ resources, in addition to the switch profiles that might have been defined. If you do not have a resource profile defined for a particular security check, and a user issues a request that would involve making that check, IBM MQ denies access. You do not need to define profiles for security types relating to any security switches that you have deactivated.

We explain the following profiles in this section:

- ▶ “Profiles for connection security” on page 136
- ▶ “Profiles for queue security” on page 136
- ▶ “Profiles for process security” on page 137
- ▶ “Profiles for namelist security” on page 137
- ▶ “Profiles for alternate user security” on page 138
- ▶ “Profiles for context security” on page 138
- ▶ “Profiles for command security” on page 139
- ▶ “Profiles for command resource security” on page 139
- ▶ “IBM MQ Security Management” on page 139

Profiles for connection security

If connection security is active, you must define profiles in the MQCONN class and permit the necessary groups or user IDs access to those profiles, so that they can connect to IBM MQ.

To enable a connection to be made, you must grant users RACF READ access to the appropriate profile. Table 5-9 shows the different connection types allowed by IBM MQ and the corresponding profile for the connection type. Users trying to connect to IBM MQ should have READ access to the corresponding profile. Profile names are of the form *hlq.XXXX*, where *hlq* is the queue manager name for queue manager level security (or queue-sharing group name for queue-sharing group level security) and *XXXX* is the connection type.

Table 5-9 Connection security RACF profiles

Connection type	RACF profile
Batch (including z/OS batch jobs, TSO applications, z/OS UNIX System Services (USS) sign-ons, and DB2 stored procedures)	hlq.BATCH
CICS connections	hlq.CICS
IMS connections from control region and application processing regions	hlq.IMS
IBM MQ channel initiator	hlq.CHIN

Profiles for queue security

If queue security is active, you must define profiles in the MQQUEUE or GMQQUEUE classes and permit the necessary groups or user IDs access to these profiles, so they can issue IBM MQ API requests that use queues.

Profiles for queue security take this form:

hlq.queue name

Where *hlq* is either the queue manager name or the queue-sharing group name, and *queue name* is the name of the queue being opened, as specified in the MQOPEN or MQPUT1 call.

The RACF access required to open a queue depends on the MQOPEN or MQPUT1 options specified. If more than one of the MQ00_* and MQPM0_* options are coded, the queue security check is performed for the highest RACF authority required.

Table 5-10 show the MQOPEN or MQPUT1 options and the corresponding access level required for those options.

Table 5-10 Access level for queue security

MQOPEN or MQPUT1 option	RACF access level required for hlq.queueuname
MQOO_BROWSE	READ
MQOO_INQUIRE	READ
MQOO_BIND_*	UPDATE
MQOO_INPUT_*	UPDATE
MQOO_OUTPUT or MQPUT1	UPDATE
MQOO_PASS_ALL_CONTEXT MQPMO_PASS_ALL_CONTEXT	UPDATE
MQOO_PASS_IDENTITY_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT	UPDATE
MQOO_SAVE_ALL_CONTEXT	UPDATE
MQOO_SET_IDENTITY_CONTEXT MQPMO_SET_IDENTITY_CONTEXT	UPDATE
MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT	UPDATE
MQOO_SET	ALTER

Profiles for process security

If process security is active, you must define profiles in the MQPROC or GMQPROC classes and permit the necessary groups or user IDs access to these profiles, so they can use IBM MQ interface requests that use processes.

Profiles for processes take this form:

hlq.processname

Where *hlq* is either the queue manager name or the queue-sharing group name, and *processname* is the name of the process being opened.

To open a process, the requesting user ID should have READ access to the hlq.processname profile.

Profiles for namelist security

If namelist security is active, you must define profiles in the MQNLIST or GMQNLIST classes and permit the necessary groups or user IDs access to these profiles.

Profiles for processes take this form:

hlq.namelistname

Where *hlq* is either the queue manager name or the queue-sharing group name, and *namelistname* is the name of the namelist being opened.

To open a namelist, the requesting user ID should have READ access to the hlq.namelistname profile.

Profiles for alternate user security

If alternate user security is active, you must define profiles in the MQADMIN class and permit the necessary groups or user IDs access to these profiles, so that they can use the ALTERNATE_USER_AUTHORITY options when the object is opened. An alternate user profile gives the requesting user ID access to resources associated with the user ID specified in the alternate user ID.

Profiles for alternate user security can be specified at subsystem level or at queue-sharing group level and take the following form:

`hlq.ALTERNATE.USER.alternateuserid`

Where *hlq* is either the queue manager name or the queue-sharing group name, and *alternateuserid* is the value of `AlternateUserId` field in the object descriptor.

The requesting user ID should have UPDATE access to the alternate user profile to enable the user of the alternate ID for resource security checks.

Profiles for context security

If context security is active, you must define a profile in the MQADMIN class called *hlq.CONTEXT.queueename*, where *hlq* can be either qmgr-name (queue manager name) or qsg-name (queue-sharing group name), and *queueename* can be either the full name of the queue you want to define the context profile for or for a generic profile.

Table 5-11 shows the access level required, depending on the specification of the context options when the queue is opened.

Table 5-11 Access level for context security

MQOPEN or MQPUT1 option	RACF access level required to hlq.CONTEXT.queueename
MQPMO_NO_CONTEXT	No context security check
MQPMO_DEFAULT_CONTEXT	No context security check
MQOO_SAVE_ALL_CONTEXT	No context security check
MQOO_PASS_IDENTITY_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT	READ
MQOO_PASS_ALL_CONTEXT MQPMO_PASS_ALL_CONTEXT	READ
MQOO_SET_IDENTITY_CONTEXT MQPMO_SET_IDENTITY_CONTEXT	UPDATE
MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT	UPDATE
MQOO_OUTPUT or MQPUT1 (USAGE(XMITQ))	CONTROL

Profiles for command security

If you want security checking for commands (so you have not defined the command security switch profile *hlq.NO.CMD.CHECKS*) you must add profiles to the MQCMD5 class.

The profile takes this form:

hlq.verb.object

Where *hlq* is the queue manager name or queue-sharing group name, *verb* is the IBM MQ script command (MQSC) verb, and *object* is the object part of the MQSC command.

Profiles for command resource security

If you have not defined the command resource security switch profile, *hlq.NO.CMD.RESC.CHECKS*, because you want security checking for resources associated with commands, you must add resource profiles to the MQADMIN class for each resource.

Profiles for command resource security checking take this form:

hlq.type.resourcename

Where *hlq* is the queue manager name or queue-sharing group name, and *resourcename* is the name of the resource that needs to be protected. The *type* part of the profile name is required to distinguish the type of resource. The values for *type* can be CHANNEL, QUEUE, PROCESS, or NAMELIST.

5.1.4 IBM MQ Security Management

IBM MQ uses an in-storage table to hold information relating to each user and the access requests made by each user.

To manage this table efficiently and to reduce the number of requests made from IBM MQ to the external security manager (ESM), these controls are available:

► User ID reverification

If the RACF definition of a user who is using IBM MQ resources has been changed (for example, by connecting the user to a new group), you can tell the queue manager to sign this user on again the next time it tries to access an IBM MQ resource. You can do this by using the IBM MQ command **RVERIFY SECURITY**.

► User ID timeouts

When a user accesses an IBM MQ resource, the queue manager tries to sign this user on to the queue manager (if subsystem security is active). This means that the user is authenticated to the ESM. This user remains signed on to IBM MQ until either the queue manager is shut down, or until the user ID is “timed out” (the authentication lapses) or reverified (reauthenticated).

When a user is timed out, the user ID is “signed off” within the queue manager and any security-related information retained for this user is discarded. The signing on and off of the user within the queue manager is transparent to the application program and to the user. Users are eligible for a timeout when they have not used any IBM MQ resources for a predetermined amount of time. This time period is set by the MQSC **ALTER SECURITY** command.

- Security refresh

When a queue is opened for the first time IBM MQ performs a RACF check to obtain the user's access rights and places this information in the cache. The cached data includes user IDs and resources on which security checking has been performed. If the queue is opened again by the same user the presence of the cached data means IBM MQ does not have to issue RACF checks, which improves performance.

The action of a security refresh is to discard any cached security information and so force IBM MQ to make a new check against RACF. Whenever you add, change or delete a RACF resource profile, you must tell the queue managers to refresh the security information that they hold.

To do this, issue the following commands:

- The RACF **SETROPTS RACLIST(classname) REFRESH** command to refresh at the RACF level. If you are using generic profiles, you must also issue the RACF **SETROPTS GENERIC(classname) REFRESH** command.
- The MQSC **REFRESH SECURITY** command to refresh the security information held by the queue manager.

- Display security status

To display the status of the security switches, and other security controls, you can issue the MQSC **DISPLAY SECURITY** command.

5.1.5 IBM MQ CICS adapter

The IBM MQ CICS adapter is required to use IBM MQ within CICS. The IBM MQ CICS adapter provides the following information to IBM MQ specifically for use in IBM MQ security:

- Whether CICS resource-level security is active for this transaction, as specified on the RESSEC or RSLC operand of the RDO TRANSACTION definition.
- CICS address space user ID and the task or terminal user ID.

The IBM MQ CICS adapter's control functions (the CKQC transaction) let you manage the connections between CICS and IBM MQ dynamically. You can invoke these functions using the IBM MQ-CICS adapter panels, from the command line, or from a CICS application. You can use the adapter's control function to perform the following tasks:

- Start or stop a connection to a queue manager.
- Modify the current connection. For example, you can reset the connection statistics, change the adapter's trace ID number, and enable or disable the API-crossing exit.
- Display the current status of a connection and the statistics associated with that connection.
- Start or stop an instance of the task initiator transaction, CKTI. (*Task initiator transaction* is CICS terminology. In IBM MQ terminology, this is a *trigger monitor*.)
- Display details of the current instances of CKTI.
- Display details of the CICS tasks currently using the adapter.

5.1.6 IBM MQ IMS adapter

The IBM MQ IMS adapter is required to use IBM MQ within IMS. The IBM MQ IMS adapter allows you to connect your queue manager to IMS, and enables IMS applications to use the IBM MQ interface.

The IBM MQ IMS adapter is the interface between IMS application programs and an IBM MQ subsystem. It makes it possible for IMS application programs to use the IBM MQ interface. The IMS adapter receives and interprets requests for access to IBM MQ using the External Subsystem Attach Facility (ESAF) provided by IMS.

The IBM MQ IMS adapter provides access to IBM MQ resources for programs running in the following modes or states:

- ▶ Task (TCB) mode
- ▶ Problem state
- ▶ Non-cross-memory mode
- ▶ Non-access register mode

The adapter provides a connection thread from an application task control block (TCB) to IBM MQ.

5.1.7 Channel security

When you are using channels, the security features available depend on which communications protocol you are going to use. If you use TCP, there are no security features provided with the communications protocol, although you can use SSL. If you are using APPC, you can flow user ID information from the sending MCA through the network to the destination MCA for verification.

For both protocols, you can specify which user IDs you want to check for security purposes and how many. Again, the choices available to you depend on which protocol you are using, what you specify when you define the channel, and the RESLEVEL settings for the channel initiator.

SSL setup

You can also use the Secure Sockets Layer (SSL) to provide channel security. SSL support is provided with IBM MQ for z/OS. SSL is an industry-standard protocol that provides a data security layer between application protocols and the communications layer, usually TCP/IP. SSL uses encryption techniques, digital signatures and digital certificates to provide message privacy, message integrity and mutual authentication between clients and servers.

5.1.8 Threats and risks

This section explains some common threats to IBM MQ security:

- ▶ Sniffing

Computers with access to a network can record the traffic flowing through it. If data or commands are sent unencrypted as IBM MQ messages, it is easy for unauthorized people to passively eavesdrop. The keyword here is *passive*. Sniffing is an activity that leaves no trace in a network log.

Sniffing is a threat to confidentiality, but if user IDs and passwords are sniffed, the threat becomes more serious because the attacker could then impersonate a legitimate user. It may be possible for an attacker to also use sniffed packets off the network and replay them. For example, if an WebSphere MQ message is an instruction to make a payment, then it is possible that the attacker can replay this message several times.

- ▶ Impersonation

The attacker tricks a security system, passing as an authorized user. There are three possible levels of impersonation:

- First, the attacker might be able to impersonate the IBM MQ administrator.
- Second, the attacker might be able to impersonate a valid queue manager and receive messages.
- Third, an attacker might be able to impersonate a sending queue manager and send messages.

- ▶ Flooding

If an attacker sends large amounts of data, it can consume the network bandwidth. If the attacker has gained access to an IBM MQ queue manager, the server can become overused, which prevents access to other users or greatly affects performance. An attacker might also be able to flood downstream queue managers by using the communication between queue managers. Flooding is a threat to availability.

- ▶ Application weakness

The TCP/IP protocol, some of its applications and some operating systems have inherent security shortcomings, sometimes due to the objectives of their original design (openness, easy communication between computers and applications).

Company-developed applications, such as or software purchased from vendors, might have security weaknesses that attackers can exploit. The degree of the damage depends on the nature of the problem. The most common damage is for a system to be shut down.

5.1.9 Event monitoring and recording

This section describes how to monitor the performance and resource use of IBM MQ. We discuss how to collect statistics of IBM MQ using SMF records and how to use IBM MQ event alerts to monitor the system.

Collecting statistics

You can use MQSC DISPLAY command to obtain information about the current state of IBM MQ. They provide information about the status of the command server, process definitions, queues, the queue manager, and so on.

You can also record performance statistics and accounting data for IBM MQ by using the IBM MQ trace facility. The data generated by IBM MQ is sent to these destinations:

- ▶ The System Management Facility (SMF), specifically as SMF record type 115, subtypes 1 and 2, for the performance statistics trace
- ▶ The SMF, specifically as SMF record type 116, subtypes zero, 1, and 2, for the accounting trace

You can start the IBM MQ trace facility at any time by issuing the **START TRACE** command. You can also start collecting some trace information automatically if you specify YES in the SMFSTAT (SMF STATISTICS) and SMFACCT (SMF ACCOUNTING) parameters of the CSQ6SYSP macro. For more information about the CSQ6SYSP macro, see the *IBM MQ for z/OS System Setup Guide*.

For daily monitoring, information is sent to SMF (the default destination). SMF data sets usually contain information from other systems; this information is not available for reporting until the SMF data set is dumped.

IBM MQ events

IBM MQ instrumentation events provide information about errors, warnings, and other significant occurrences in a queue manager. You can monitor the operation of all your queue managers by incorporating these events into your own system management application. These instrumentation events fall into the following categories:

- ▶ Queue manager events

These events are related to the definitions of resources within queue managers. For example, an application attempts to put a message to a queue that does not exist.

- ▶ Performance events

These events are notifications that a threshold condition has been reached Monitoring performance and resource use by a resource. For example, a queue depth limit has been reached, or the queue was not serviced within a predefined time limit.

- ▶ Channel events

These events are reported by channels as a result of conditions detected during their operation. For example, when a channel instance is stopped.

- ▶ Configuration events

These events are notifications that an object has been created, changed or deleted.

When an event occurs, the queue manager puts an event message in the appropriate event queue, if defined. The event message contains information about the event that can be retrieved by a suitable IBM MQ application. IBM MQ events can be enabled using the MQ commands or the operations and control panels. Channel events can be disabled only by altering the definition of the event queue to PUT(DISABLED).

5.2 Guiding principles for configuring security

In this section, we review preferred practices for securing IBM MQ and cover the following topics:

- ▶ Common misconfigurations
- ▶ Security considerations

5.2.1 Common misconfigurations

It is important to be aware of the two most common IBM MQ security-related misconfigurations.

Improper switch settings

RACF switch profiles are a powerful feature to switch on or off security checking or various resources. See “IBM MQ RACF switch profiles” on page 129 for details of RACF switch profiles. Because a single RACF profile can turn certain security features on or off, it is possible to easily misconfigure them.

It is a good practice to check status of the switch profiles. You can do this by checking the JESMSGLG of the queue manager’s MSTR address space.

RESLEVEL access for channel initiator

If the user ID of the Channel Initiator address space has an access level of NONE to RESLEVEL, two user IDs are checked for all channels. However, if the Channel Initiator user ID has an access level of CONTROL or ALTER to RESLEVEL, no user IDs are checked for MQI access by any channels.

Unfortunately, it’s easy to misconfigure access to RESLEVEL for the Channel Initiator, with the result that security checks are bypassed for all channels. The preferred practice is to make sure that the Channel Initiator’s user ID has an access level of NONE to RESLEVEL. This causes two user IDs to be checked. For more information about setting the RESLEVEL profile, see 5.1.2, “IBM MQ RACF RESLEVEL profile” on page 133.

Some sites grant their administrators CONTROL or ALTER access to RESLEVEL, because they assume that their administrators need access to all queues. Depending on the sensitivity of the data being transported by the queue manager, this assumption is not always a sound one, because on z/OS, IBM MQ administrators can define queues without also being granted access to the messages they contain. Also, if administrators have CONTROL or ALTER access to RESLEVEL, then no AUDIT records are created to track access done through the MQI by these users.

Given that most companies want (or are required) to scrutinize activities of privileged users, this practice might not be consistent with local security policies. It’s also important to make sure that the Channel Initiator’s user ID is not added to the Administrators’ RACF group, because that disables MQ interface security for channels. Connecting the Channel Initiator’s user ID to the RACF group that was created for administrators, when that group has been granted CONTROL or ALTER access to RESLEVEL, is a common error that essentially turns off MQ interface security for all channels.

5.2.2 Security considerations

There are several security exposures that seem to be common to many IBM MQ installations. These often stem from default options that are accepted by during IBM MQ installation and seldom changed or reviewed:

- Naming conventions

Use proper queue naming conventions on z/OS to simplify RACF profiles. For example, use some kind of company.dept.application hierarchy, so you can assign access to company.*, company.dept.*, or company.dept.application*. Avoid common mistakes, such as putting the queue type at the start of the queue name, for example, queue local (QL) or queue remote (QR), as in these examples:

- QL.MYLOCALQ
- QR.MYREMOTEQ

- Delete default items

The IBM MQ installation process places many default objects on a system that can later on allow attacks on the infrastructure. These should all be deleted or protected before moving to a production environment. Use of objects that start with SYSTEM.DEFAULT should be carefully examined. If you do not know why an object is there, rename it and determine whether the system requires having it back. If it doesn't, delete it.

- Protect command server queues

Command server queues, such as SYSTEM.COMMAND.INPUT on z/OS, are gateways into an IBM MQ infrastructure that should be protected as much as any other major electronic gateway into an enterprise.

- Change the CMDUSER on z/OS

The default user ID for command security checks is CSQOPR. We suggest that you change this (in CSQ6SYSP) and give restricted authority to the ID. See the *IBM MQ for z/OS System Setup Guide* for more information about the CSQ6SYSP macro.

- Blank user IDs

RACF allows user IDs to be blank. Access to these user IDs should be carefully controlled. The default security timeout value is 54 minutes, which is large. Consider setting this to a shorter time interval to minimize the security risk.

- Generic profiles

Defining too many generic profiles for RACF or not setting up RESLEVEL correctly is costly in terms of system maintenance. Therefore, carefully plan what resources need protection and how you will protect them.

- ISPF panel access

Access to the ISPF panels available for IBM MQ for z/OS should be controlled to ensure that objects cannot be defined, altered, started, or stopped without adequate authority.



Session management

In this chapter we provide a security concepts and architecture overview for IBM Session Manager for z/OS. Then, we describe some guiding principles for configuring session management security.

This chapter covers the following topics:

- ▶ IBM Session Manager basics
- ▶ Security concepts and architecture
- ▶ Guiding principles for configuring security

6.1 IBM Session Manager basics

IBM Session Manager for z/OS (ISM) is a Systems Network Architecture (SNA) or TCP/IP session manager that offers secure access to multiple z/OS systems from a single 3270 terminal or equivalent. It gives users a single secure sign-on, and allows controlled access to all of their applications from multiple concurrent virtual sessions using a wide range of tools and facilities.

The 3270 terminal session can exist on either an SNA or TCP/IP network. The terminal session can simultaneously connect to and navigate between multiple SNA and TCP/IP applications. Session Manager delivers greater user productivity, enhanced system use, increased security, reduced costs, and improved communication.

Session Manager offers both internal or external security to grant users access to servers and applications through dynamic or, static menus through online administration.

External security managers (ESMs), such as Resource Access Control Facility (RACF), allow dynamic menus to be automatically built based on access to RACF resource groups that the user is authorized to access.

Internal security allows static menus to be configured by security administrators and may vary for each user. Users with similar job roles might not have access to the same servers and applications of other users though the Session Manager. Menus offered to each user group or role may be inconsistent and require additional access to be defined through online administration (OLA).

Automatic logon to servers and applications can be achieved using certificate express signon using a temporary, one-time-use PassTicket, which is to be validated against RACF with no need for the users to enter their passwords multiple times.

6.2 Security concepts and architecture

In this section, we cover user authentication, dynamic menus, online administration, and the setup of security to support the environment in the following sections:

- ▶ 6.2.1, “User authentication” on page 150
- ▶ 6.2.2, “Static menus” on page 156
- ▶ 6.2.3, “Security setup” on page 158
- ▶ 6.2.4, “Session Manager commands” on page 162
- ▶ 6.2.5, “Session Manager command statements” on page 163
- ▶ 6.2.6, “Threats and risks” on page 163
- ▶ 6.2.7, “Event monitoring and recording” on page 164

IBM Session Manager has various components that each need to be protected to ensure integrity within the environment is maintained. These components include configuration files, security exits, and user IDs and passwords.

Figure 6-1 shows the main components of Session Manager and the flow from an interactive user session through to access to remote networks on which applications and servers reside.

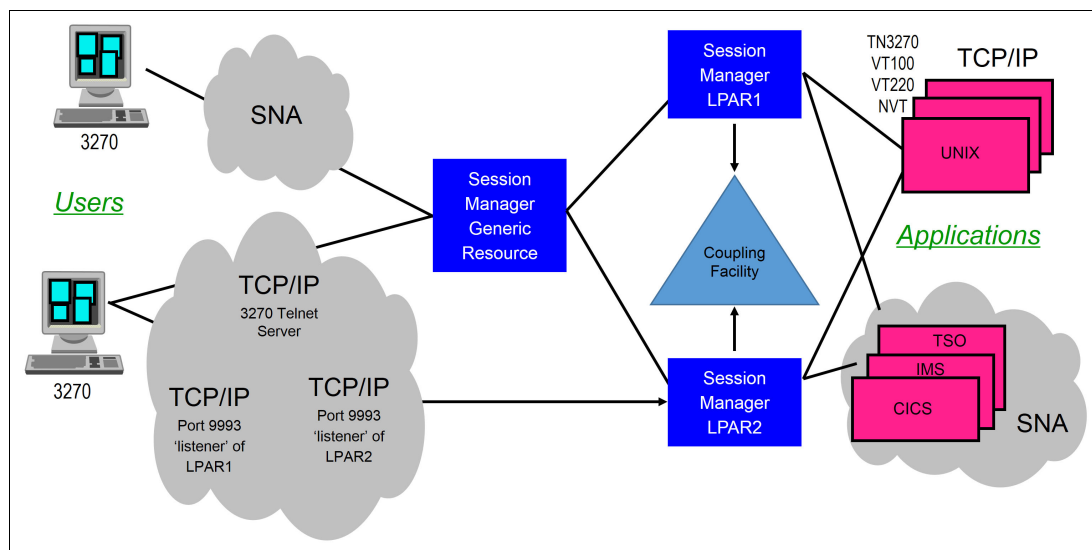


Figure 6-1 General Session Manager architecture

This figure shows that the user has several possibilities to log on to session manager which is primarily a single front end of z/OS based applications. Mostly, these are SNA applications, such as CICS, IMS, TSO, and so on. If users want direct access to TCP/IP related systems, such as UNIX, they can simply select defined UNIX sessions from the menu.

To set up a high availability system you can use the generic resource concept which is available in the communication server. The Session Manager is used in combination with the coupling facility as shown in Figure 6-1. In this way, users always have a front end available, regardless of which Session Manager is active. The coupling facility determines whether the user is already logged on at another Session Manager. If the user is reconnecting, it routes the user to the correct target system.

In addition to the standard connection over a Telnet 3270 server, the Session Manager has its own Telnet server implemented over the standard port 9993, through which the user can log on. The user sees the same look and feel as the standard Telnet 3270 server when using it.

Users can access Session Manager through either a SNA or TCP/IP connection, typically from their local PC using emulation software, such as PCOMM or several other emulators that are available.

The Session Manager then makes a logical connection from the user's PC to one or more remote servers or applications through either SNA or TCP/IP connectivity.

Regardless of how they connect initially, the connection to the Session Manager enables users to easily access a range of servers and applications concurrently without the need to configure access to each session individually.

Jump key: Users can jump between sessions by using a configured jump or function key.

An automated logon with a startscript automatically logs on with the user's ID and password to the specific sessions. The password is encoded and stored and is not readable. If a PassTicket is to be implemented and used, the startscripts can then use this PassTicket instead of the password. We discuss this concept further under "PassTickets" on page 158.

6.2.1 User authentication

When a users sign on to the Session Manager front end, they need to be self-authenticated by entering their user IDs and passwords. If the password intervals are reached, they might need to change their passwords within the Signon Panel, as shown in Figure 6-2.

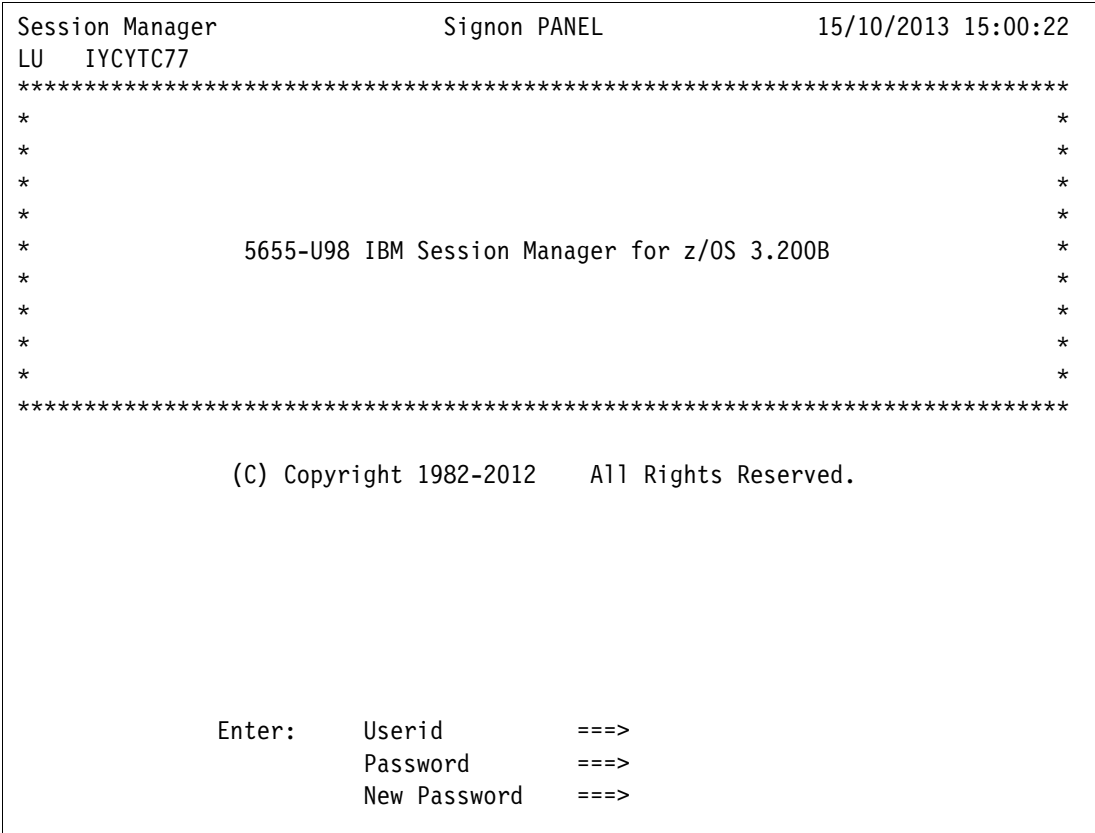


Figure 6-2 IBM Session Manager sample Signon Panel

There are two ways that users can be authorized to access remote servers and applications, depending on how you choose to use the Session Manager and the existence of an external security manager. You can choose between these options:

- Using passwords specified on the USER statement

This is the least secure method of controlling access to Session Manager.

Where internal security is used, there is a password field on the Session Manager USER statement.

The USER statement: The USER statement is the internal definition of the user. This statement determines what authorization users have and what kind of sessions they will see if the static menu is used.

- External security manager

This is the preferred method. Using one or more ESMs such as RACF, to authenticate users, and to determine a user's Session Manager capabilities.

Figure 6-3 shows the flow for two users who are logging on to Session Manager:

- User a logging on using a traditional user ID and password
- User b logging on using a PassTicket using the certificate express login facility, user b.

This diagram shows common session managers and the resulting menu presented to the user which allows them to access one or more servers and applications for which they are authorized.

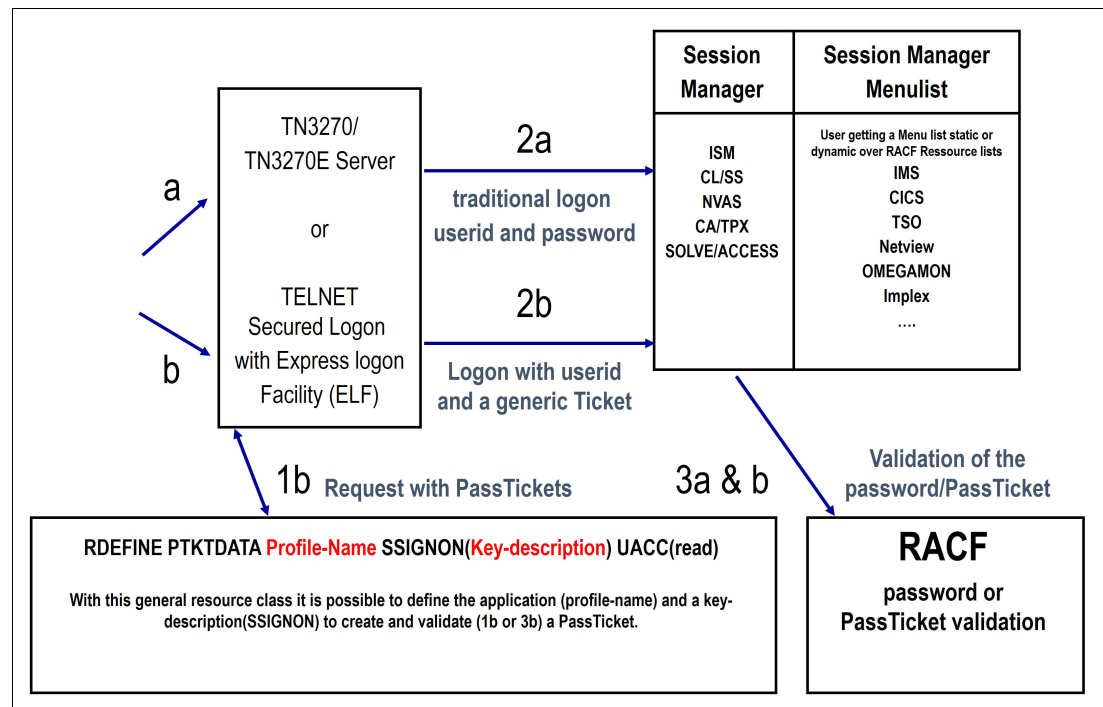


Figure 6-3 User logon using traditional, and using certificate express login

Take a closer look at the flow details:

- Following path a, the user may access the Session Manager using the traditional logon method using a user ID and password as shown by 2a.
- Following path b, the certificate express logon facility generates a temporary PassTicket by interacting with the external security manager as defined by the RACF resource class PTKTDATA. This PassTicket is then used in 2b to log on to the Session Manager.
- Following flow through to end of this flow, authorization for a user to access the Session Manager is determined, and they are presented with a list of servers and applications to choose from.
- In 3a, you see the validation of the user's password and in 3b the validation of the PassTicket by RACF.

Dynamic menus

Dynamic menus are built based on a user's authority to RACF resource class profiles and their authority to these resources through a group permission, or where the user is explicitly granted read access. The default RACF class is the FACILITY class. However, this can be any class suitable for your local installation. Resources within the defined RACF class have no prefix by default. However, this can be configured to use a resource name that is more meaningful to security administrators.

Figure 6-4 shows the Session Manager parameters that can be configured for using dynamic menus. In this figure, we see how Session Manager can be configured to use a customized resource class and resource name rather than the defaults.

Session Manager - Admin		SECURITY Sub Parameter list		04/10/2013 08:48:15	
LU IYCYTC50		Name of entry: ISZSYS02		CS1VSAM	
Sub Parameter:		Setting (SYSTEM or inherited)			
AUTHRESNAME					
AUTHCLASSNAME					
OLARESNAME					
DYNMLOG NO					
DYNMAUTSTHID NO					
DYNMCLASS \$ISMCLASS(Default is FACILITY)					
DYNMDROPSESSION NO					
DYNMHIDE YES					
DYNMLOGMAX 0					
DYNMRESNM ISZ.APPL.					
DYNMTYPE APPL					
ESMPRFACC NO					
ESMPRFCLNM \$ISMCLASS(Default is nothing)					
ESMPRFRSNM ISZ.PROF.					
PASSPHRASE					
SIGNONACCESS NO					
SIGNONCLASS					
SIGNONRESNAME					
TERMINALACCESS NO					
TERMINALCLASS					
TERMINALRESNAME					
TYPECLASSGLOBAL NO					
COMMENT					
Command ==>					
Available options: S - Select (default) D - Delete R - Reset H - Help					
PF1 Help	2	3 End	4 Actmsg	5	6
PF7 Backward	8 Forward	9	10	11	12 Cancel

Figure 6-4 Administration view showing configuration of resource class and resource name

Note: We use the RACF \$ISMCLASS resource class and ISZ.APPL.<application name/APPL> resource name to show choices for configuring IBM Session Manager.

Next, we describe the configuration shown in Figure 6-4 on page 152:

- ▶ Each application name (APPL) defined for access in the default PROFILE is checked against the external security system resource profiles configured for your installation.
- ▶ The RACF resource profile is then checked to determine whether access is permitted or denied to the user.
- ▶ Each application can be *un-hidden* by specifying an APPL to which the user must have READ access. Otherwise, access is denied and the menu hides this option. Examples:
 - ISZ.APPL.TSO CLASS (FACILITY) ID (ISMUSER) ACCESS (READ)
 - ISZ.APPL.CICSTOR1 CLASS (FACILITY) ID (ISMUSER) ACCESS (NONE)

Group authorization

Design RACF group structures to logically group user access based on job role, function or even clients.

A well-thoughtout approach to creating this structure improves the experience of the users. Dynamic menus are built that enable them to access all common servers, applications, or clients and hide any that they are not entitled or authorized to access.

The design of security using RACF groups authorized to Session Manager resource class profiles should also take into consideration the servers, applications, and clients that are common and are typically used concurrently by a user.

Where specific access to a server or application should not be granted to a group, explicit access can be used to further control access for users to various servers and applications.

Create RACF statements to define resources, groups, and permissions for these groups, allowing users to be authorized to access the server or application through the dynamic menu made available to them.

Authorization to servers and applications are shown as resources in Figure 6-5 within a resource class. This figure shows the three methods a user can be authorized to access one or more resources defined to the resource class.

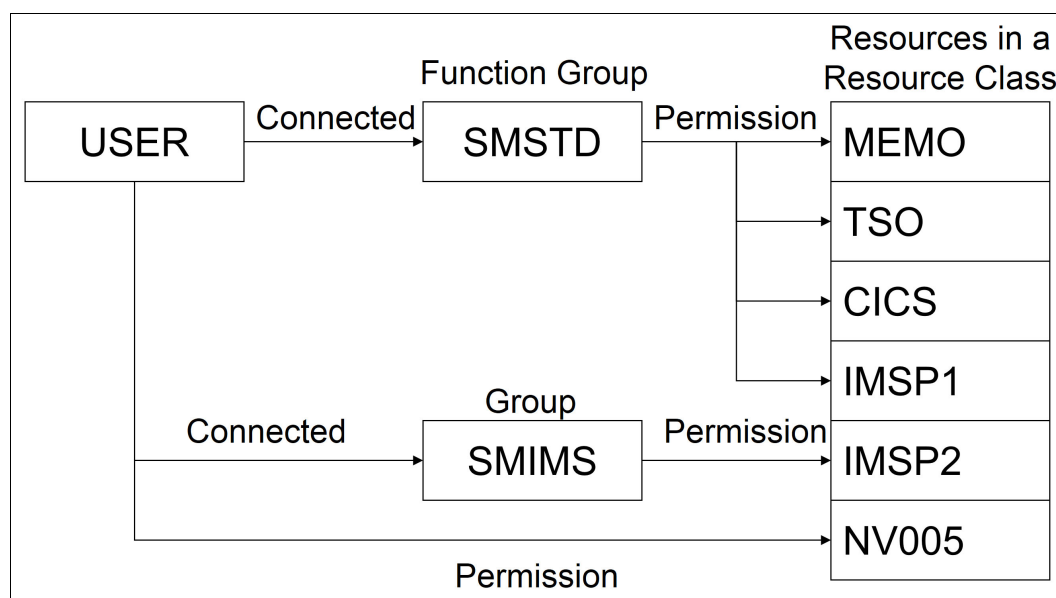


Figure 6-5 Overview of user connections to function groups and groups

Users may be connected in any of three ways:

- **Function group**

A user connected to a function group is permitted to access more than one resource in a resource class. This function group is associated with a group of users who typically perform the same tasks as other members of the group to access multiple servers and applications.

- **Group**

A user connected to a group is permitted access to a single resource in a resource class.

- **Direct permission**

The user is given specific access to a resource, but not as part of a function group or group membership.

Direct permissions: The use of direct permissions is not convenient and requires a RACF database refresh before the permission becomes active for the user.

It is also helpful to have a meaningful naming convention for the group names. Two possible conventions are proposed in Table 6-1 and Table 6-2 however these can also be defined to align with current conventions used within your organization:

- **Convention one**

The eight character name is defined as shown in Table 6-1.

Table 6-1 First suggested naming convention

Character position	Recommendation
1	# or \$ character
2	A single letter. Use P for Production, T for Test, or F for Function.
3 to 6 inclusive	Four letters for the session ID
7 to 8 inclusive	A two-digit sequence number

- **Convention two**

The eight character name is defined as shown in Table 6-2.

Table 6-2 Second suggested naming convention

Character position	Recommendation
1	# or \$ character, or a special non-numeric sign
2 to 6 inclusive	Five letters for the session ID. If the session ID name is too long, it may be abbreviated. If the session ID name is shorter than five letters, fill the remaining space with X characters.
7 to 8 inclusive	A two-digit sequence number

Figure 6-6 shows the basic concept for authorizing users to servers and applications through permissions to RACF resources.

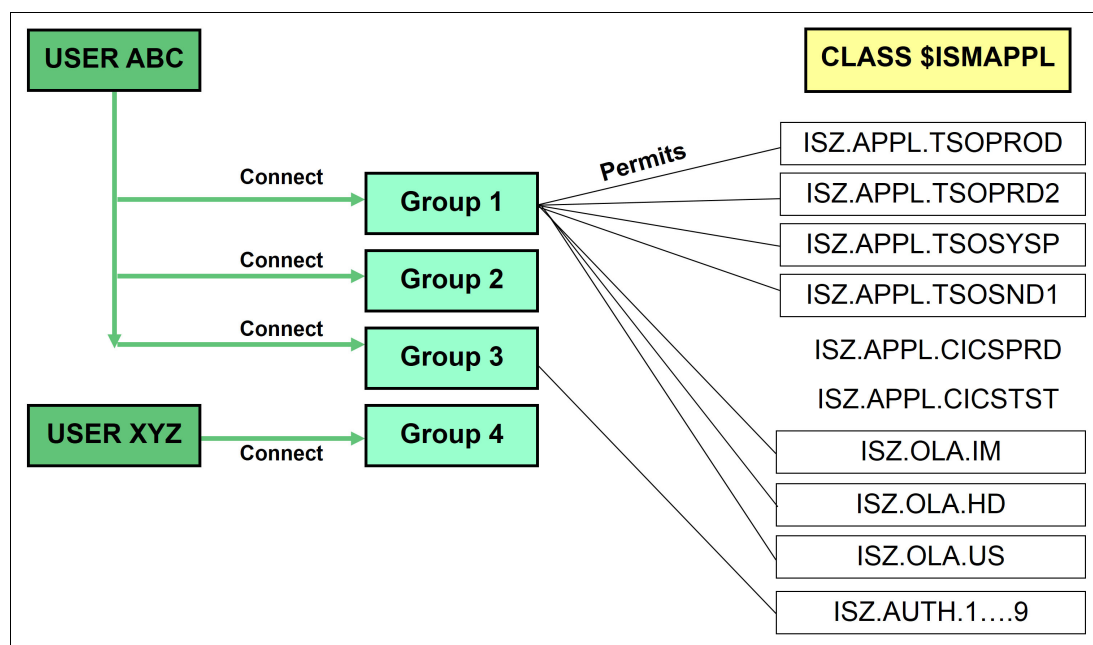


Figure 6-6 RACF group concept for a dynamic menu

In this figure, user ABC is connected to several groups:

- ▶ Group 1 allows the user to access several TSO servers whilst not allowing access to others.
The user is not authorized to access CICSPRD or CICSTST.
- ▶ Group 2 does not authorize the user to any servers or applications accessible through the Session Manager and may simply be authorizing them to other resources such as application data sets.
- ▶ Group 3 allows the user to access a function group that may not typically be made available to all users with a similar role through the Session Manager.
The resources ISZ.OLA.* in this figure are related to the roles that are available to control access for the Session Manager online administration as discussed further in “OLA classes” on page 157.
- ▶ Group 4 shows user XYZ with no authority to use any servers or applications through the Session Manager.
Group 4 has no permissions to any of the resources, unless explicitly authorized.

Figure 6-7 shows a sample Session Manager view with limited servers and applications made available to the user through generation of dynamic menus using RACF group authorizations as the illustration shows.

Session Manager		Menu2 PANEL			15/10/2013 15:35:03	
LU	IYCYTC77				USER ABC	
Se1	Description	APPLID	Status	ACB	LOGMODE	NODE
---	-----	-----	-----	---	-----	----
PF1	TSO on LPAR PROD	TSOPROD	AVAIL			
PF2	TSO on LPAR PRD2	TSOPRD2	AVAIL			
PF3	TSO on LPAR SYSP	TSOSYSP	AVAIL			
PF9	TSO in SANDBOX	TSOSND1	AVAIL			
800	OLA Administration	OLA	AVAIL			
HARDCOPY =						
ISZ4012I Signon complete for ABC IP address 10.10.10.10 PORT 49653						
====>						

Figure 6-7 Dynamic menu after user ABC has successfully logged on

6.2.2 Static menus

Static menus are manually configured for each user and present a user with a default set of menu options.

The servers and applications that are available to the user through the menu are limited to servers and applications that they are authorized to use.

Internal security

Internal security is configured and maintained manually for each user through online administration (OLA).

Although the user might be presented with a menu that appears to allow access to a server or application, this does not necessarily mean that the user is authorized to access that server or application through the Session Manager.

Review internal security to maintain an accurate list of servers and applications that are available to the user to limit the menu options available to them.

Online administration

Online administration allows menus to be authorized to individual users through an interface to data set configuration.

The objective is to distinguish between system administration and user requirements. In meeting the objective, the best approach is to provide the minimum possible authority for individual users.

User-based security allows the administrators to perform the following tasks:

- ▶ Define implementers, administrators, users, and so on
- ▶ Configure access to servers, applications, clients and so on for users
- ▶ Add, modify, delete, and display configuration for users
- ▶ Define what gets displayed

OLA classes

Using the OLAClass attribute, authority for users may be limited based on the functions that they should be authorized to perform, as shown in Table 6-3.

Table 6-3 OLA classes for users

OLA class	Category	Description
AD	Administrator	Allows access to all areas of OLA but with the restriction from modifying OLA security settings
BT	Batch administrator	Allows access to all areas of OLA and required for batch administration
IM	Implementer	Allows access to all areas of OLA
LA	Local administrator	Allows access to all areas of OLA but with the restriction from modifying OLA security settings or system settings
SU	Super User	User has access to all of the 'my user definition' settings. Password and trace are not displayed, and user cannot modify the OLAClass or AUTH parameter
US	User	Restrict access to all areas of OLA except common user parameters
NO	No access	Restrict access to all areas of OLA.

Further information about OLA classes can be found in *Session Manager for z/OS V3.1 Online and Batch Administration*, SC34-7149-00.

6.2.3 Security setup

In this section, we explain the security setup for IBM Session Manager including the following factors:

- ▶ PassTickets
- ▶ Certificate express logon
- ▶ Session Manager scripts
- ▶ Security exits

PassTickets

Using a temporary PassTicket for applications such as those in the following list eliminates the need for a user to log on by using a user ID and password:

- ▶ TSO
- ▶ IBM CICS
- ▶ IBM IMS
- ▶ IBM Tivoli NetView® for z/OS
- ▶ IBM Tivoli OMEGAMON® XE on z/OS

RACF can issue temporary PassTickets which have a lifespan of 10 minutes. These are used by Session manager to log the user on automatically to the requested server or application.

The variable in which the generated PassTicket is to be passed to the application must be specified in the PTKTDATA parameter.

For several applications the entry in PTKTDATA is just the VTAM application name. Some, especially third-party products, have their own fixed naming rules for this variable.

Certificate express logon

The certificate express logon facility uses user certificates to automate and reduce the effort of logging on or transferring between servers and applications. This eliminates the need for users to log on by using a traditional user ID and password to access each server.

The use of certificates requires each user to be issued with a personal client certificate on token or smart card which is validated by the external security Manager RACF using the following Secure Sockets Layer (SSL) methods:

- ▶ One-to-one certificate to user ID association
- ▶ Certificate name Filtering
- ▶ HostIDMappings certificate extension

These 3 methods are deeper explained in the *z/OS V1R12.0 Security Server RACF Security Administrator's Guide*, SA22-7683-14 within the following chapters:

- ▶ Enabling client login using Certificates
- ▶ Using a hostIDMappings extension

The purpose of a certificate is to assure a program or a user that it is safe to allow the proposed connection and, if encryption is involved, to provide the necessary encryption/decryption keys. They are usually issued by certificate authorities (CAs), which are organizations that are trusted by the industry as a whole and who are in the business of issuing digital certificates. A CA's certificate, which is also known as a root certificate, includes the CA's signature and a validity period, among other things.

Encryption and authentication are performed by means of a pair of keys, one public and one private. The public key is embedded in a certificate, known as a site or server certificate. The certificate contains several items of information, including the name of the CA that issued the certificate, the name and public key of the server or client, the CA's signature, and the date and serial number of the certificate. The private key is created when you create a self-signed certificate or a CA certificate request and is used to decrypt messages from, and encrypt messages to clients.

You must register all workstation client certificates with RACF using the RACDCERT command. This associates the certificates with the ID of the user who is attempting to log on.

Session Manager scripts

We now look at the Session Manager scripts and the processing flow from a user logon to completion of the process.

Here is the flow for this process:

- ▶ The EXIT21 (sign on validation) exit

The first check, after the user has entered their user ID and password, is validation of the user's credentials provided at the signon window. This validation is done by the ISZEXT21 assembled exit.
- ▶ The EXIT22 (signon completion) exit

This exit establishes the number of sessions that are available to the user and generates the dynamic menu based on authority granted through the external security system.

This shows users only those servers and applications that they are authorized to access.

The dynamic menu is created by assembled ISZEXT22 exit.
- ▶ The EXIT31 (slave session pre-initiation) exit

When the user is selecting a session where a PassTicket must be generated, the assembled ISZEXT31 exit takes control if this has been implemented.

Preinstalled exits: For IBM Session Manager, the exits ISZEXT21 and ISZEXT22 are pre-installed.

Table 6-4 lists available exists that can be used based on how your installation chooses to configure Session Manager.

Table 6-4 Summary of exits used by Session Manager

Exit	Exit script	Description
ISZE00	Not applicable	Multiple exit driver
ISZE21	E21	At signon validation
ISZE22	E22	Signon completion
ISZE31	E31	Slave session pre-initiation

See Chapter 3 in *IBM Session Manager Implementation*, SG24-6459 for detailed information about each exit.

Multiple exit driver: If the ISZEXT00 exit is coded in the CONFIG parameter, the driver exit loads all of the other modules, starting with ISZEXTxx. See *Session Manager for z/OS Facilities Reference*, SC34-6297, for a complete description of dynamic menus and the multiple exit driver.

Protecting exit source code is as important as protecting the security exits themselves to prevent a user from possibly circumventing any controls within the environment.

Access to view or modify source must be on an as needed basis to prevent unauthorized users from modification, and introducing exits that might circumvent the underlying security checking though RACF.

Security exits

Session Manager provides a user exit facility that allows administrators a great deal of flexibility and control within the Session Manager environment. Exits are available at critical processing paths to allow the administrator to have granular control.

User exits are provided for installation and customizing, and may be either of these programs:

- ▶ Assembler programs
- ▶ COBOL programs

For each exit, there is a corresponding script exit, and the two work as a pair. The E21 exit pair follows these naming conventions:

- ▶ E21, exit script
- ▶ ISZEXT21, the assembled and linked Assembler or COBOL code exit

The script exit is invoked first and, based on a specific return code from this script, the assembled code might or might not be called.

You do not need to code both the script and assembled exit and you might need to have only the script exit in place.

The flow of script processing is shown in Figure 6-8.

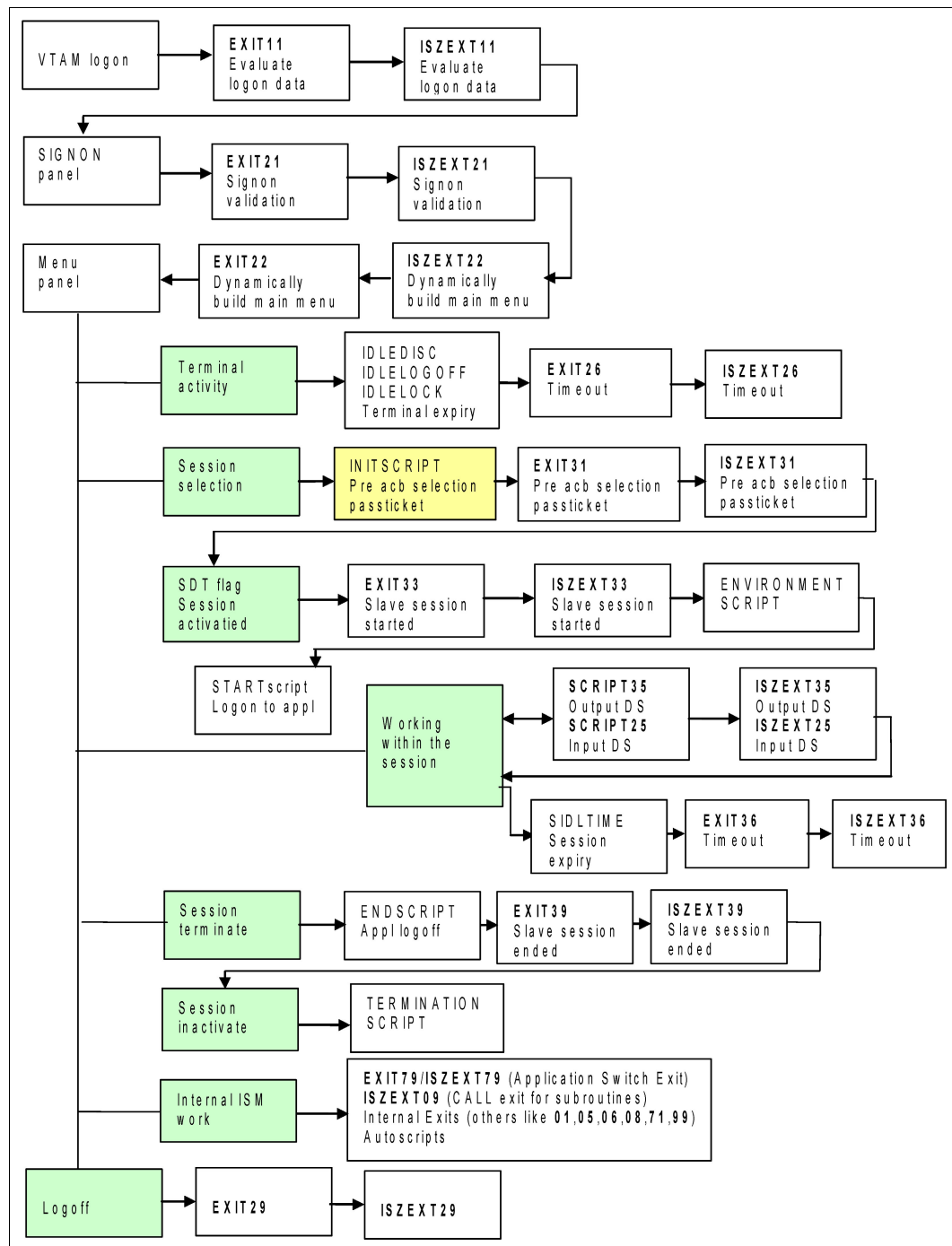


Figure 6-8 Session Manager script processing flow

6.2.4 Session Manager commands

Each Session Manager command has a security code allocated to it. The security code is a numeric digit from 1 to 9, where 1 is the least secure and 9 is the most secure. There is a default security code associated with each command. It may be overwritten by an installation by using the COMMAND statement in the Session Manager configuration.

With the implementation of the ISZEXT22 exit, the users see the sessions on their menus when they have read authority for a specific resource.

As Figure 6-4 on page 152 shows, this can be achieved by any of the following parameters or a combination of them:

- The APPL name is configured in Session Manager parameters:

```
DYNMCLASS    $ISMCLASS
DYNMRESNM    ISZ.APPL.<application name/APPL>
```

- The profile is assigned by the security system through the following commands:

```
ESMPRFCLNM $ISMCLASS
ESMPRFRSNM ISZ.PROF.<profile>
```

- Both can be combined, with profiles checked first and then the application checked to confirm whether the user has Read permission to access the application.

- These parameters add security and administrative levels:

AUTH	Manages the command authorization rights within the Session Manager.
PASS	Manages the Passphrase.
SIGNON	Manages who can log on to the Session Manager. This can also be achieved using the RACF class APPL.
TERMINAL	Which terminals have access to the Session Manager. This can also be achieved using the RACF protection.

When defining the AUTHCLASNAME and AUTHRESNAME with RACF, profiles such as ISZ.AUTH.1 to ISZ.AUTH.9, which give a user Read access to ISZ.AUTH.1, allow the user to issue only commands that are associated with those resources, as shown in Table 6-5 on page 163.

If you give the user the read access to ISZ.AUTH.9, which includes all commands, the user will be able to perform all tasks, including basic and administrator tasks, as shown in Table 6-5 on page 163.

Resource name prefixes: If you want to use prefixes, such as ISZ.APPL.* or similar ones, ensure that the RACF class includes this prefix and that the length of the resource name is supported.

The command types default to 1 for basic and 9 for administrators, as shown in Table 6-5. For more information about each command, see *IBM Session Manager for z/OS Technical Reference manual*, SC34-2805-00.

Session Manager commands can be grouped by job role or function, as shown in Table 6-5.

Table 6-5 Session Manager commands type

Command type	Security code	Commands
Basic	1	ADDSESS, BACKWARD, BRECEIVE, BWD, CONCEAL, CONFIRM, CUTEND, CUTSTART, DELSESS, DISCONNECT, DOWN, END, FILTER, FIND, FORWARD, FWD, HALTSCRIPT, HARDCOPY, HCOPTION, HELP, PASTESTART, PCTTRANSFER, PULL, QACTUSER, QQUIT, QUERY, QUIT, QUSER, RESET, RETRIEVE, RETURN, REVEAL, RIGHT, SE, SEND, SME, SPYOFF, STARTSC, TOP, TRANSFER, TTPSL, UP, VIEW, WINDOWS
Operator	5	BLOCK, DEMO, DLOG, FLASH, PLAYDS, PLAYHEX, PLAYIMAGE, RECORD, REPLAY, SPY
Administrator	9	BROADCAST, CLOSEDOWN, DELETE, DSTORE, DTERM, DUMP, FORCE, GFS, ISZTEST, PASSFREE, PUPDATE, QTASK, REMOVEUSER, SECFRESH, SPIN, STARTCP, STARTLINK, STOP, STOPACB, STOPLINK, STOPTCP, SWITCHPLX, TERMINATE, TRACE, UPDATE,

6.2.5 Session Manager command statements

Partitioned data set extended (PDSE) files used by the Session Manager need to be protected to allow access to only authorized administrators of Session Manager. See the IBM Redbooks publication titled *Best Practices Migrating to IBM Session Manager*, REDP-4156 for more information about the following Session Manager PDSE data sets:

- ▶ User
- ▶ Profile
- ▶ Terminal
- ▶ Appl
- ▶ System

6.2.6 Threats and risks

During conversion or setup of a new installation, protect any configuration command statements stored in PDSE data sets for online administration to ensure integrity.

Universal access controls, or public access to RACF resources, needs to be configured so that no default authority will be granted by setting the Universal Access Control (UACC) to NONE, using the parameter UACC(NONE) on RACF rules.

Existing Session Manager products can have complete terminal masking and assignment tools. These are used to provide stronger terminal security. Virtual Terminal Masking (VTM) may also be used for terminal security. To perform high-security tasks, an application might require that virtual terminal 01 be used. The user must then access that virtual terminal to perform the task.

When preparing for migration to Session Manager, analyze all of the VTM rules currently in use to ensure that they will be maintained. For any new installation, consider reviewing all system, application profile, and user level terminal requirements.

6.2.7 Event monitoring and recording

Recording of RACF activity is controlled and recorded in SMF logs based on auditing values set for the various RACF resources.

Set auditing values to ensure that the required level of auditing is recorded so that subsequent review and monitoring can be achieved.

SMF Records can be written by implementing the following Assembler Exits:

ISZE39SM	Writing SMF 242 records when user's session is ending
ISZE25AT	Writing SMF 242 records when a user is pressing any key in an application
ISZE33AT	Writing SMF 242 records when a user's session is starting an application
ISZE39AT	Writing SMF 242 records when a user's session closes an application

Logon and logoff can be mapped by the Session Manager audit job log or with the terminal records which are written by TCPIP or Telnet within SMF types 118 or 119.

In general, if changes are done by the administrators it is recorded in the audit job log of the started task and, if customized, in the SYSLOG as shown by the following sample messages to capture a user, system or profile update:

ISZ2030I	ABC LU PT01 Entry XYZ modified in DD USER
ISZ0247I	PUPDATE processing DD USER Member XYZ

6.3 Guiding principles for configuring security

We now look at guiding principles for configuring security for IBM Session Manager. We look at a best approach for implementing Session Manager which will benefit the user, provide greater control over access to servers and applications, and reduce administration time.

6.3.1 Certificate express logon and PassTickets

To eliminate the risks associated with disclosure of a user password, consider certificate express logon and the use of a PassTicket or passphrase.

Consider the use of a PassTicket and passphrase as a more secure solution, rather than using traditional logon using a user ID and password. This is achieved by using exit E31. The use of a PassTicket is the preferred method, because it minimizes the risk of sensitive information being disclosed.

Ensure that any certificates used meet your security needs and if creating certificates, be sure to carefully protect any private keys.

Certificates supporting the use of a PassTicket and passphrase need to be managed so that access is removed as soon as possible after continued business need can no longer be justified.

6.3.2 Security considerations

The best approach for IBM Session Manager is to use dynamic menus to control the list of applications and servers available to a user using exits E21 and E22.

Granting users authority via a group to access the applications and the Session Manager resource APPL together presents the user with only the applications that they are authorized to access. This can be easily managed by the administrators by implementing group authority to both and granting users access through a group connection.

RACF limitations

Ensure that the RACF security database has sufficient free space to accommodate any bulk load of access rules and can accommodate future growth.

Currently, 5957 is the maximum number of users who may be connected to a group. Use of universal group can overcome this problem however, visibility to view all group members cannot be achieved through RACF directly.

Maximum group connections: Although limited to 5957 users being connected to a group, this can be reviewed by extracting members of this group from the RACF database or by using Security zSecure.

Auditing of source, exits, and configuration data sets must be in place to allow follow-up and investigation into any unusual or suspicious activities.

6.3.3 VTAM dump considerations

A VTAM dump will contain information relating to a user and their encrypted password.

Preventing unauthorized users from creating VTAM dumps that contain sensitive information will minimize any risk of exposing user password.

6.3.4 Protection of IBM Session Manager libraries and PDSE data sets

Libraries containing source code and assembled exits are to be stored in Authorized Program File (APF) libraries.

Protect Session Manager product data sets and user configuration files stored in Session Manager PDSE data sets by using standard RACF protection, limiting access to security administrators and system support personnel.



Scheduling systems

In this chapter, we provide a security concepts and architecture overview for IBM Tivoli Workload Scheduler for z/OS. Then, we describe guiding principles for configuring security.

This chapter covers the following topics:

- ▶ Tivoli Workload Scheduler for z/OS basics
- ▶ Security concepts and architecture
- ▶ Guiding principles for configuring security

7.1 Tivoli Workload Scheduler for z/OS basics

The Tivoli Workload Scheduler for z/OS is used to process scheduled work for multiple applications running in a pre-defined order once all prerequisite requirements have been met.

Tivoli Workload Scheduler for z/OS consists of two components: The controller and the tracker-started tasks.

There are three main triggers or events that can initiate work to be run by the scheduler:

- ▶ Scheduled tasks and batch jobs
- ▶ Receipt of input files from remote servers
- ▶ Automation

Product name and nicknames: Tivoli Workload Scheduler for z/OS is often called “TWS for z/OS” informally. It is also commonly referred to in the industry as “TWSz” for Tivoli Workload Scheduler for z/OS and “TWSd” for distributed environments.

7.2 Security concepts and architecture

The Workload Scheduler can have multiple instances running on any given server to support policy requirements within the enterprise.

In this section, we cover the following topics:

- ▶ 7.2.1, “Protecting the Workload Scheduler subsystem” on page 169
- ▶ 7.2.2, “Controlling access to Workload Scheduler” on page 169
- ▶ 7.2.3, “Event-triggered tracking” on page 172
- ▶ 7.2.4, “Security exits” on page 172
- ▶ 7.2.5, “Threats and risks” on page 173
- ▶ 7.2.6, “Event monitoring and recording” on page 173

There are two main components of the workload scheduler that should be considered when protecting the environment, which we discuss in more detail later in this chapter:

- ▶ The Tivoli Workload Scheduler for z/OS product needs to be protected
- ▶ Source code

Tivoli Workload Scheduler for z/OS is authorized to submit jobs on behalf of another user through the job-submit exit (EQQUX001). This grants the required level of access to data and resources, which might be of a sensitive nature.

7.2.1 Protecting the Workload Scheduler subsystem

The default Resource and Access Control Facility (RACF) IBMOPC class is used by the workload scheduler to protect functions, resources, and retain a reliable database of schedule work to be processed. If multiple controller subsystems require separate policies, they require separate classes be defined to RACF and activated.

AUTHDEF is defined in the member of the EQQPARM library as specified by the PARM parameter on the JCL EXEC statement using the CLASS parameter. For more information about setting this, see *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning*, SC32-1265-08.

Consider the following checklist when using another RACF class:

- ▶ The resource class must be defined in the RACF class descriptor and routing tables.
- ▶ IBMOPC is a predefined class that you can use with no need for an IPL, if only one class is required.
- ▶ After a RACF migration, consider redefining any class you defined in a previous version of RACF.
- ▶ The default class OPCCLASS is not already defined in RACF. Before using this class, make sure there are the necessary entries in the RACF class descriptor and routing tables.

Tivoli Workload Scheduler for z/OS name: Tivoli Workload Scheduler for z/OS was known as Operations Planning and Control (OPC) before its rebranding in 2001.

7.2.2 Controlling access to Workload Scheduler

Security involves three levels of protection:

- ▶ The Tivoli Workload Scheduler for z/OS subsystem determines whether a user can establish communication with Tivoli Workload Scheduler for z/OS.
- ▶ Fixed resources protect functions in Tivoli Workload Scheduler for z/OS (for example, modifying the current plan or requesting a backup of a resource data set).
- ▶ Subresources protect Tivoli Workload Scheduler for z/OS data.

The level of access given to a user at one level determines the default access that the user has at remaining levels. For example, if a user has update access to the Workload Scheduler subsystem, then that user has, by default, update access to all fixed resources. A user's access to fixed resources, in turn, determines the default access to subresources. The default value is used when a resource has not been specifically protected. Common functions within the workload scheduler can be controlled by restricting access to functions based on job role or function.

You can use fixed resources and subresources to protect Workload Scheduler functions and data. Fixed resources are always checked as part of the Workload Scheduler dialog. Subresources are checked only if they are defined in the AUTHDEF statement.

It is best that you *do not* grant access to individual users. Instead, try to group users into different categories. You can then define a RACF user group for each category of users. With RACF user groups, you do not need to change access lists of different profiles as often. When you must make a change, you add or remove a user ID in the group or move the user ID to another group.

These categories are used in many Tivoli Workload Scheduler installations:

- ▶ Schedulers
- ▶ Workstation operators
- ▶ Workload Scheduler shift leaders
- ▶ Machine room operators
- ▶ Workload Scheduler system support

Also consider using generic profiles when specifying RACF resource names. Resources protected by generic profiles have similar names and identical security requirements.

Table 7-1 describes all fixed resources and subresources. Use the table to determine which resources you should define to RACF.

Table 7-1 Protected Fixed Resources and Subresources

Fixed resource	Subresource	RACF resource name	Description
AD	AD.ADNAME AD.ADGDEF AD.NAME AD.OWNE AD.GROUP AD.JOBNAME AD.SECLEM AD.UFVAL	AD ADA.name ADD.name ADN.name AD0.name ADG.name ADJ.name ADM.NAME ADU.field_name.field_value	Application-description file Application name Group definition ID name Operation extended name in application description Owner ID Authority group ID Operation job name in application description Security element name User field name and value
CL	CL.CALNAME	CL CLC.name	Calendar data Calendar name
CP	CP.ADNAME CP.CPGDEF CP.NAME CP.OWNER CP.GROUP CP.JOBNAME CP.WSNAME CP.ZWSOPER CP.SECLEM CP.UFVAL CP.RESNAME	CP CPA.name CPD.name CPN.name CP0.name CPG.name CPJ.name CPW.name CPZ.name CPM.name CPU.field_name.field_value CPR.RESNAME	Current plan file Occurrence name Occurrence group definition ID Operation extended name Occurrence owner ID Occurrence authority group ID Occurrence operation name Current plan workstation name Workstation name used by an operation Security element name Operation user field name and value Special resource name
ET	ET.ETNAME ET.ADNAME	ETT ETE.name ETA.name	ETT dialog Name of triggering event Name of application to be added
JS	JS.ADNAME JS.OWNER JS.GROUP JS.JOBNAME JS.WSNAME	JS JSA.name JS0.name JSG.name JSJ.name JSW.name	JCL and job library file Occurrence name Occurrence owner ID Occurrence authority group ID Occurrence operation name Current plan workstation name
JV	JV.OWNER JV.TABNAME	JV JV0.name JVT.name	JCL variable definition file Owner ID of JCL variable definition table Name of JCL variable table

Fixed resource	Subresource	RACF resource name	Description
LT	LT.ADNAME LT.LTGDDEF LT.OWNER	LT LTA. <i>name</i> LTD. <i>name</i> LTO. <i>name</i>	Long-term plan file Occurrence name Occurrence group-definition ID Occurrence owner ID
OI	OI.ADNAME	OI OIA. <i>name</i>	Operator instruction file Application name
PR	PR.PERNAME	PR PRP. <i>name</i>	Period data Period name
RD	RD.RDNAME	RD RDR. <i>name</i>	Special resources file Special resource name
RG	RG.RGNAME RG.OWNER	RG RGY. <i>name</i> RGO. <i>name</i>	Run cycle group Run cycle group name Run cycle group owner
RL	RL.ADNAME RL.OWNER RL.GROUP RL.WSNAME RL.WSSTAT	RL RLA. <i>name</i> RLO. <i>name</i> RLG. <i>name</i> RLW. <i>name</i> RLX. <i>name</i>	Ready list data Occurrence name Occurrence owner ID Occurrence authority group ID Current-plan workstation name Current-plan workstation changed by WSSTAT
RP	RP.REPTYPE	RP RPT. <i>reptype</i>	Dynamic Workload Console reports report type, depending on the report that you request: <ul style="list-style-type: none"> ▶ RUNHIST for job run history reports ▶ RUNSTATS for job run statistics ▶ WWR for workstation workload runtimes reports ▶ WWS for workstation workload summary ▶ SQL for reports obtained by customized SQL queries
SR	SR.SRNAME	SR SRS. <i>name</i>	Special resources in the current plan Special resource name
WS	WS.WSNAME	WS WSW. <i>name</i>	Workstation data Workstation name in workstation database
ARC		ARC	Activate or deactivate automatic recovery
BKP		BKP	Request backup of a resource data set
BUL		BUL	Initiate bulk discovery for the monitoring agent
CMAC		CMAC	Cleanup action
CONT		CONT	Refresh RACF subresources
ETAC		ETAC	Activate/deactivate event-triggered tracking
EXEC		EXEC	EX (execute) row command
JSUB		JSUB	Activate/deactivate job submit
REFR		REFR	Refresh LTP and delete CP
WSCL		WSCL	All-workstations-closed data

For more information, see *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning*, SC32-1265-08.

Subresource verification: The subresource name and the RACF resource name are not the same. You specify the subresource name shown in Table 7-1 on page 170, Column 2, by using the SUBRESOURCES keyword of the AUTHDEF member to start subresource verification. The corresponding RACF resource name shown in Column 3 must be defined in the general resource class that is used by the Workload Scheduler, specified by the CLASS keyword of AUTHDEF.

7.2.3 Event-triggered tracking

Event-triggered tracking (ETT) adds occurrences to the current plan based on a triggering event. You can use ETT to track work that has been submitted outside of the Workload Scheduler, or simply to respond to an on-demand request for processing.

Work triggered through ETT can include these examples:

- ▶ Recycling of subsystems, following an IPL
- ▶ Submission of batch job on receipt of a remote file transfer

As work can be triggered to be run by the Workload Scheduler on receipt of a remote file transfer, restrictions should be in place to limit any adverse processing of data.

Any authorized attempt to transfer false or even empty data to the host server could trigger work to be processed by the Workload Scheduler with unknown results.

The authority of FTP user IDs and the scope of file transfers any one FTP user ID is authorized to receive should be limited to a minimum. This can limit the ability for multiple false inputs to enter batch processing with successor tasks triggered to run automatically.

7.2.4 Security exits

The Workload Scheduler provides a user exit facility that allows administrators a great deal of flexibility and control within the Workload Scheduler environment. Exits are available at critical processing paths to allow the administrator to have granular control.

Sample assembler user exits are provided for installation customization. Table 7-2 lists available exits that can be used based on how your installation chooses to configure the Workload Scheduler.

Table 7-2 Summary of exits used by Tivoli Workload Scheduler

Exit	Description
EQQUX001	Sample Job submit exit
EQQUX002	Sample job-library-read exit
EQQUX004	Sample event-filtering exit
EQQUX011	Sample job-tracking log write exit
EQQUX013	Sample job-tailoring prevention exit

For more detailed information about each exit, see *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning*, SC32-1265-08.

Protecting exit source code is as important as protecting the security exits themselves to prevent a user from circumventing the controls within the environment.

Access to view or modify source must be on an as needed basis to prevent unauthorized users from modifying and introducing exits that may circumvent to underlying security checking though RACF.

RUSER: The EQQUX001 exit sets the RUSER value under which a process is executed. This is described in more detail in 7.3.2, “Submitting user IDs” on page 174.

7.2.5 Threats and risks

With the authority of the Workload Scheduler address space, there is a risk that support personnel will have unauthorized access to possibly view or modify sensitive information.

7.2.6 Event monitoring and recording

Recording of RACF activity is controlled and recorded in SMF logs based on auditing values set for the various RACF resources.

Auditing values should be set to ensure that the required level of auditing is recorded so subsequent review and monitoring can be achieved.

7.3 Guiding principles for configuring security

When building a schedule within the Workload Scheduler, consideration should be given to logically break down and group together common tasks or functions to allow greater control over security within the enterprise.

In this section, we cover the following topics:

- ▶ 7.3.1, “Workload Scheduler applications” on page 173
- ▶ 7.3.2, “Submitting user IDs” on page 174
- ▶ 7.3.3, “Protecting JES resources” on page 175
- ▶ 7.3.4, “Batch and protection of source JCL and code” on page 176

7.3.1 Workload Scheduler applications

Within the Workload Scheduler, batch jobs can be grouped to run under logical applications, groups, departments, locations, and so on with unique batch user IDs.

The Workload Scheduler started task is to be granted surrogate authority to submit jobs on behalf of the various batch user IDs. Authority can be granted to users through group connections to the various outputs.

7.3.2 Submitting user IDs

Consider what should be visible to various individuals. Batch process and related output can be owned by logical owners and not visible to all users of the Workload Scheduler.

The outputs from these processes can further be restricted to individual groups or users who have a business need for access. We discuss this in more detail under 7.2.3, “Event-triggered tracking” on page 172.

How to determine authority

You can determine the authority given to a job or started task in several ways:

- ▶ You can submit work with the authority of the Workload Scheduler address space. The job or started task is given the same authority as the controller or tracker that has a Submit subtask that submits the work. Work that is transmitted from the controller and then submitted by the tracker is given the authority of the tracker.
- ▶ Another method is to use the job submit exit, EQQUX001. This exit is called when the Workload Scheduler is about to submit work.

You can use the RUSER parameter of the EQQUX001 exit to cause the job or started task to be submitted with a specified user ID. The RUSER name is supported even if the job or started task is first sent to a tracker before being started.

In certain circumstances, you might need to include a password in the JCL to propagate the authority of a particular user. You can use the EQQUX001 exit to modify the JCL and include a password. The JCL is saved in the JCL repository (JSn) data set before the exit is called, which avoids the need to store JCL with specific passwords. This method prevents the password from being visible externally. For more information about the job submit exit, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Rather than using the user ID of the started task, a batch should be scheduled to run under various batch user IDs, with the Workload Scheduler authorized to submit work on their behalf. Each of these user IDs is authorized to perform a limited number of tasks or functions, with explicit authority to various parts of the environment.

Consider surrogate job submission to authorize jobs submitted by the Workload Scheduler for by specifying the started task user ID be added as a surrogate user for each of your systems and production batch IDs.

Example 7-1 shows the started task user ID OPCSTC being authorized to submit jobs on behalf of the batch user ID ACCT001.

Example 7-1 Authorization for the Workload Scheduler to submit work.

```
REDEF SURROGAT ACCT001.SUBMIT UACC(NONE) OWN(STCGRP)
PERMIT ACCT001.SUBMIT CLASS(SURROGAT) ID(OPCSTC) ACC(READ)
```

Through the EQQUX001 user exit, the Workload Scheduler is authorized to submit work on behalf of the nominated user ID or, where not specified, the user ID running the Workload Scheduler started task.

The RUSER parameter

In this section, we describe the various ways that the RUSER parameter can be set by using the EQQUX001 exit, as mentioned in 7.2.4, “Security exits” on page 172.

The RUSER parameter of the EQQUX01 user exit determines the name of the RACF user ID that owns the job. This parameter contains eight blanks when the exit is called. If required, the exit can update this parameter to cause the job to be submitted under the specified user ID.

The returned RUSER value is stored in the CP file. It guarantees that whenever a stand-alone cleanup job is submitted, it has the same owner as the original job.

Different ways to set it are suggested in the EQQUX001 sample job in the SEQQSAMP library. A code sample for extracting the value of the USER parameter from the JOB card is included also. See the comments in the sample for more information.

You can also use the RUSER parameter to modify the user ID that is submitting a job with a centralized script. If you use this keyword to specify the name of the user who submits the specified script or command on a Microsoft Windows fault-tolerant workstation, you can perform these tasks:

- ▶ Associate this user name with the Windows workstation in the USRREC initialization statement
- ▶ Set LOCALPSW(YES) in the TOPOLOGY statement, and then use the user utility to define the name and password of the user in a local file on the Windows workstation

This parameter is supported even if the job is sent to another system via a submit/release data set. So, there is a possibility that the SUBMIT SUBTASK of the controller or of the tracker that is submitting a given job might abend while executing under that RUSER-supplied user ID, rather than under the user ID associated with the Tivoli Workload Scheduler for z/OS started task. If this occurs, DUMPTASK might fail with an ABEND913 if the user ID in control does not have WRITE access to the SYSMDUMP data set. For this reason, SYSMDUMP data sets should be defined with a UACC of UPDATE. That is, they should be WRITE-ENABLED to all user IDs under that an Tivoli Workload Scheduler for z/OS scheduled job might possibly be submitted.

If RUSER is blank and the job card does not specify the USER keyword, the job is submitted with the authority of the Tivoli Workload Scheduler for z/OS started task.

7.3.3 Protecting JES resources

Access to JES output should be carefully considered to prevent unauthorized persons from viewing the output generated by work managed through the Workload Scheduler.

Support personnel who need visibility to job output to be able to diagnose problems and restart jobs may also have access to view output from the job, including customer reports, financial data, and so on.

RACF resource protection for output should be configured to separate the role of support and owner. Not limiting this access could allow support personnel to view sensitive or other information for which they have no justification.

The RACF command in Example 7-2 demonstrates how to grant access to support personnel to limit visibility to only those components of the JES output relevant to recovering the batch issue.

Example 7-2 Grant access using the RDEFINE RACF command

```
RDEFINE JESSPOOL &&RACLNDE.<userid>.*.*.*.JES* OWNER(<owner>) UACC(None)
```

This is typically used in conjunction with RACF commands to allow read access to only those who are authorized by explicitly restricting access with a Universal Access Control (UACC) of none, as in Example 7-3.

Example 7-3 Restricting access via the RACF RDEFINE command

```
RDEFINE JESSPOOL &&RACLNDE.<userid>.** OWNER(<owner>) UACC(None)
```

Protecting data sets

For basic security of the Workload Scheduler data, restrict access to all product data sets.

Two categories of users need different levels of access to the product data sets:

- ▶ **Software support**
People who must be able to debug problems and reorganize VSAM files. You might give them alter access to all the product data sets.
- ▶ **Administrators and operators**
These groups must be able to use the product dialogs. They need read access to ISPF-related data sets (such as the panel and message libraries), but they do not access the databases (such as the workstation database) directly. Those files are accessed by the Workload Scheduler subsystem, not by any code in the TSO user's address space. Authority to access the data for a dialog user is given using the authorization functions provided by the product.

The tasks started by Workload Scheduler need the following conditions:

- ▶ *Alter* access to VSAM data sets
- ▶ *Read* access to input data sets, such as the message library (EQQMLIB) and the parameter library (EQQPARM).
- ▶ *Update* access to all other Tivoli Workload Scheduler for z/OS data sets
- ▶ *Update* access to catalogs and *alter* access to data sets for all work in that workload
- ▶ *Scheduler* tracks if you use the Restart and Cleanup functions

7.3.4 Batch and protection of source JCL and code

Production source code, JCL, and inputs should be protected from unauthorized alteration to ensure the integrity of batch processing is maintained.

Note: Protection of these resources is *critical* so that no unauthorized processing can take place.

The JCL comes from the JS file (EQQJSnDS), the JCL job library (EQQJBLIB), or the job-library-read exit (EQQUX002).

JCL libraries used by Tivoli Workload Scheduler for z/OS are defined in the DD statement that is defined in the product's started task, as shown in Example 7-4.

Example 7-4 DD statement identifying the JCL libraries

```
//EQQJBLIB DD DISP=SHR,DSN=<concatenated library list>
```

Without controls in place to protect source JCL, changes could be made that might then enter the environment through the Workload Scheduler, allowing unauthorized processing. This could lead to confidential or sensitive information being exposed, copied, modified, and so on. For this reason, it is critical to protect source JCL libraries containing production batch jobs, parmlib members, or any component of a batch job that could enter the environment and be processed by Tivoli Workload Scheduler for z/OS that is authorized to submit work on behalf of another user.

In addition to this, any components relating to automation should also be considered carefully, because that could lead to unauthorized activity taking place. As a simple example, a job could be modified to copy a single data set to another user's library at some point in the batch process without affecting that process. This small change could be undetectable without secondary controls to review what appears to be authorized activity.

After this, the user might then be authorized to the production data, which the unauthorized user could then access outside of the Tivoli Workload Scheduler for z/OS environment, or be able to access the data through other authorized ways. The job could then be modified back to its original state, possibly covering the unauthorized access to data. This data can then be processed offline easily in an insecure environment, which can potentially lead to a breach in security, disclosure of private and sensitive information, and so on.

Authority to access these libraries should be limited to support personnel, including batch schedulers and operators.

Where a user is authorized to migrate test code into a production environment, the activity might not be captured if auditing is not set to capture successful update or write activities. Setting the auditing value to capture all successful attempts to update JCL and source code can assist in investigating any unusual activity.

Related publications

The publications listed in this section are considered particularly suitable for more detailed information about the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Some publications referenced in this list might be available in softcopy only.

- ▶ *Security Functions of IBM DB2 10 for z/OS*, SG24-7959
- ▶ *IBM Virtualization Engine TS7700 Release 1.4a: Tape Virtualization for System z Servers*, SG24-7312
- ▶ *z/OS Identity Propagation*, SG24-7850
- ▶ *WebSphere Application Server V7 Messaging Administration Guide*, SG24-7770
- ▶ *WebSphere Application Server V7.0 Security Guide*, SG24-7660
- ▶ *Best Practices Migrating to IBM Session Manager*, REDP-4156

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, on the following website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



SG24-8196-00

ISBN 0738441023

Printed in U.S.A.

Get connected

