

# ABCs of IBM z/OS System Programming Volume 6

Security on z/OS

RACF and SAF

Cryptography



Karan Singh  
Rui Feio  
Oerjan Lundgren  
Bob McCormack  
Rita Pleus  
Paul Rogers

**Red**books





International Technical Support Organization

**ABCs of IBM z/OS System Programming Volume 6**

August 2014

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

## **Second Edition (August 2014)**

This edition applies to Version 1, Release 7 of z/OS (5694-A01), Version 1 Release 7 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

**© Copyright International Business Machines Corporation 2008, 2014. All rights reserved.**

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

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
Authors .....	ix
Now you can become a published author, too! .....	x
Comments welcome .....	xi
Stay connected to IBM Redbooks .....	xi
<b>Chapter 1. Introduction to IBM z/OS security</b> .....	1
1.1 z/OS basic security facilities .....	2
1.2 z/OS Security Server components .....	4
1.3 Integrated Security Services .....	5
1.4 Cryptographic Services .....	6
1.5 Security Level 3 .....	8
1.6 IBM Tivoli Directory Server for z/OS .....	8
<b>Chapter 2. System Authorization Facility</b> .....	9
2.1 SAF overview .....	10
2.2 SAF in detail .....	12
<b>Chapter 3. IBM z/OS Security Server RACF</b> .....	17
3.1 What is RACF? .....	18
3.2 RACF functions .....	19
3.3 RACF ISPF panel .....	22
3.4 RACF profiles .....	23
3.5 RACF commands .....	25
3.6 User authentication .....	27
3.7 Resource managers .....	28
3.8 RACF classes .....	30
3.9 Security administration with RACF .....	31
3.10 RACF user identification and verification .....	33
3.11 RACF user profile .....	34
3.12 RACF user attributes .....	36
3.13 RACF user segments .....	38
3.14 RACF user ID and password .....	41
3.15 Adding a new user to RACF .....	43
3.16 Reset a user password .....	44
3.17 Alter a user ID .....	46
3.18 Change a user password interval .....	47
3.19 Delete a user ID .....	48
3.20 User-related RACF commands .....	49
3.21 RACF groups .....	50
3.22 RACF group structure example .....	52
3.23 RACF group-related commands: Add a group .....	53
3.24 RACF group-related commands: Alter a group .....	54
3.25 RACF group-related commands: Delete a group .....	55
3.26 Connect a user to a group .....	56
3.27 Remove a user from a group .....	57

3.28	Data sets and general resources . . . . .	58
3.29	More on profiles for data sets and general resources . . . . .	59
3.30	Data set profiles . . . . .	61
3.31	Defining data set profiles. . . . .	63
3.32	Data set profile access list . . . . .	65
3.33	Add a data set profile . . . . .	67
3.34	Alter a data set profile . . . . .	68
3.35	Search RACF database using a mask . . . . .	69
3.36	Data set-related commands . . . . .	70
3.37	Data set-related commands, continued . . . . .	71
3.38	General resources-related commands . . . . .	72
3.39	General resources-related commands, continued . . . . .	73
3.40	General resources-related commands, continued . . . . .	74
3.41	SET RACF system options . . . . .	75
3.42	Statistic-related options. . . . .	76
3.43	Password-related options . . . . .	78
3.44	Data set-related options . . . . .	80
3.45	Class-related options . . . . .	83
3.46	Authorization checking-related options . . . . .	86
3.47	Tape-related options . . . . .	88
3.48	RVARY and other options for initial setup. . . . .	90
3.49	RACF and auditing . . . . .	93
3.50	Auditor-related options . . . . .	95
3.51	SETROPTS: Display options (LIST) . . . . .	97
3.52	RACF monitoring. . . . .	98
3.53	RACF monitoring, continued. . . . .	99
3.54	RACF monitoring, continued. . . . .	100
3.55	RACF auditing tools . . . . .	101
3.56	RACF auditing - IRRADU00 . . . . .	102
3.57	RACF auditing: RACF Report Writer. . . . .	104
3.58	RACF auditing: DSMON . . . . .	106
3.59	RACF auditing: IRRDBU00. . . . .	114
3.60	RACF and Dynamic CDT . . . . .	116
3.61	RACF and protecting the program . . . . .	117
3.62	RACF remote sharing facility . . . . .	121
3.63	RRSF nodes and modes. . . . .	122
3.64	RRSF: User ID associations . . . . .	123
3.65	RRSF: Command direction . . . . .	124
3.66	RRSF over TCP/IP . . . . .	125
3.67	RACF and interaction with other subsystems . . . . .	126
3.68	RACF and digital certificates - the authentication problem . . . . .	135
3.69	Overview of digital certificates. . . . .	138
3.70	Certificate use in z/OS . . . . .	140
3.71	Management of certificates by RACF . . . . .	142
 <b>Chapter 4. Integrated Security Services . . . . .</b>		<b>155</b>
4.1	Introduction to Kerberos . . . . .	156
4.2	Kerberos terminology . . . . .	157
4.3	Kerberos protocol overview. . . . .	158
4.4	Get a ticket-granting ticket . . . . .	159
4.5	Request a service ticket . . . . .	162
4.6	Authenticate to target server. . . . .	163
4.7	Kerberos inter-realm trust relationship . . . . .	165

4.8	Some assumptions to Kerberos	166
4.9	Implementing Network Authentication Service	167
4.10	Setting up the Kerberos environment variable files	170
4.11	Setting up HFS for Kerberos cache files	173
4.12	Kerberos integrated with RACF	174
4.13	Define Kerberos local principals	178
4.14	Define Kerberos foreign principals	181
4.15	Kerberos user commands	182
4.16	Auditing	193
4.17	Overview of EIM	194
4.18	EIM concepts	196
4.19	Setting up EIM in z/OS	207
4.20	Installing and configuring EIM on z/OS	209
4.21	Domain authentication methods	214
4.22	EIM additional administration tasks	216
4.23	RACF support for EIM	223
4.24	Storing LDAP binding information in a profile	224
4.25	Setting up a registry name for your local RACF registry	225
4.26	Introduction to Open Cryptographic Enhanced Plug-ins	229
4.27	How OCEP works with a security server	230
<b>Chapter 5. Cryptographic Services</b>		<b>231</b>
5.1	Introduction to cryptography	232
5.2	Cryptographic capabilities	233
5.3	Symmetric and asymmetric encryption algorithms	234
5.4	Symmetric encryption algorithms	235
5.5	Asymmetric encryption algorithms	236
5.6	Use of cryptosystems: Data privacy	237
5.7	Use of cryptosystems: Data integrity	239
5.8	Use of cryptosystems: Digital signatures	240
5.9	IBM Common Cryptographic Architecture	242
5.10	IBM System zEC12: Cryptographic overview	245
5.11	CP Assist for Cryptographic Functions	247
5.12	Crypto Express4S feature	248
5.13	Crypto Express3 feature	250
5.14	DES key management	251
5.15	DES encryption	253
5.16	DES key forms	254
5.17	Key distribution: Key export	256
5.18	Key distribution: Key import	257
5.19	PKA key management	258
5.20	Integrated Cryptographic Service Facility	262
<b>Related publications</b>		<b>265</b>
IBM Redbooks publications		265
Other publications		266
How to get IBM Redbooks publications		267





# 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](http://www.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®	NetView®	System z®
CICS®	OS/390®	Tivoli®
DB2®	OS/400®	VTAM®
Domino®	Parallel Sysplex®	WebSphere®
eServer™	PowerPC®	z/Architecture®
GDPS®	RACF®	z/OS®
Geographically Dispersed Parallel Sysplex™	Redbooks®	z/VM®
IBM®	Redbooks (logo)  ®	z9®
IMS™	Resource Measurement Facility™	zEnterprise®
Language Environment®	RMF™	zSecure™
MVS™	S/390®	
	System z9®	

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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.

# Preface

The ABCs of IBM® z/OS® System Programming is an 11-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you want to become more familiar with z/OS in your current environment or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection can serve as a powerful technical tool.

Following are the contents of the volumes:

- ▶ Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
- ▶ Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, IBM Language Environment®, and SMP/E
- ▶ Volume 3: Introduction to DFSMS, data set basics, storage management hardware and software, VSAM, System-managed storage, catalogs, and DFSMSStvs
- ▶ Volume 4: Communication Server, TCP/IP, and IBM VTAM®
- ▶ Volume 5: Base and IBM Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), and IBM Geographically Dispersed Parallel Sysplex™ (IBM GDPS®)
- ▶ Volume 6: Introduction to security, IBM RACF®, digital certificates and public key infrastructure (PKI), Kerberos, cryptography and IBM z9® integrated cryptography, Lightweight Directory Access Protocol (LDAP), and Enterprise Identity Mapping (EIM)
- ▶ Volume 7: Printing in a z/OS environment, Infoprint Server, and Infoprint Central
- ▶ Volume 8: An introduction to z/OS problem diagnosis
- ▶ Volume 9: z/OS UNIX System Services
- ▶ Volume 10: Introduction to IBM z/Architecture®, IBM System z® processor design, System z connectivity, logical partition (LPAR) concepts, hardware configuration definition (HCD), and Hardware Management Console (HMC)
- ▶ Volume 11: Capacity planning, performance management, Workload Manager (WLM), IBM Resource Measurement Facility™ (RMF™), and System Management Facilities (SMF)

## Authors

This book was produced by a team of specialists from around the world working at the IBM International Technical Support Organization (ITSO), Poughkeepsie Center.

**Karan Singh** is a Project Leader with the ITSO, Poughkeepsie Center.

**Rui Feio** is an IT Specialist working at IBM Portugal. He has six years of experience in the IBM MVS™, IBM OS/390®, and z/OS fields. He provides support to IBM customers in Portugal. His areas of expertise include RACF, DFSMS, JES2, TSO, MVS, and UNIX System Services. He holds a BSc in Computer Science.

**Oerjan Lundgren** joined IBM in 1969 and has focused on performance and security-related topics. Oerjan was on assignment in Poughkeepsie for three years during the 1980s and has since participated in a number of IBM Redbooks® publication projects. Since 2000, Oerjan has been working for Pulsen Systems AB, which is an IBM Business Partner in Sweden, as a senior consultant in infrastructure design projects. Oerjan frequently teaches WLM and RMF workshops for ITSO around the world and also all System z related courses for customers as well as for universities.

**Bob McCormack** is a Senior Software Engineer at the IBM Australian Development Laboratory and has worked on many z/OS and IBM z/VM® products in a wide variety of capacities. He has a Bachelor of Applied Science degree from the University of Technology, Sydney and is an IBM Certified IT Specialist. He joined IBM in 2007 after many years with IBM Business Partners.

**Rita Pleus** is a Senior IT Specialist in IBM Global Services in IBM Germany. She has IT experience since 1986 in a variety of areas, including systems programming and operations management. Before joining IBM in 2001, she worked for a German IBM S/390® customer. Rita holds a degree in Computer Science from the University of Applied Sciences in Dortmund. Her areas of expertise include z/OS, its subsystems, and systems management. She was one of the authors of *ABCs of z/OS System Programming Volume 3*, SG24-6983.

**Paul Rogers** is a Consulting IT Specialist at the ITSO, Poughkeepsie Center, and has worked for IBM for 39 1/2 years. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS, JES3, Infoprint Server, and z/OS UNIX. Before joining the ITSO 19 1/2 years ago, Paul worked in the IBM Installation Support Center in Greenford, England, providing OS/390 and JES support for IBM EMEA and in the Washington Systems Center in Gaithersburg, Maryland.

Thanks to Paola Bari, ITSO, Poughkeepsie Center, for contributions to this project.

Thanks to the authors of the IBM Redbooks publication, System z Cryptographic Services and z/OS PKI Services:

- ▶ Jonathan Barney
- ▶ Jean Marc Darees
- ▶ Pekka Hanninen
- ▶ Robert Herman
- ▶ Guillaume Hoareau
- ▶ Patrick Kappeler
- ▶ Nikhil V Kapre
- ▶ MuHyun Kim
- ▶ Gerard Laumay
- ▶ Joel Porterie
- ▶ Vicente Ranieri Jr.
- ▶ Dominique Richard
- ▶ Daniel Turkenkopf

Thanks to Gregory P. Boyd, Advanced Technical Support, IBM, for his comments.

## 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](http://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 found at:

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

- ▶ Send your comments in an email to:

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

- ▶ Mail your comments to:

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

## 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 IBM z/OS security

In today's on demand environment, downtime is both unwelcome and costly. If your applications are not consistently available, your business can suffer. IBM System z, along with IBM software provide a comprehensive set of products and solutions to help address specific business resiliency needs and to help protect your data, transactions, and the reputation of your business.

With estimates of over 80% of corporate data residing or originating on mainframes, security and data integrity are on top of the list of critical business requirements. Thus, organizations need to deliver advanced security features with an array of user identification, authentication, auditing, and administration capabilities, combined with advancements in data encryption, intrusion detection, and overall system integrity. These capabilities are designed to sustain customer-facing, high-volume transaction rates at high service levels.

In this book, we explain how IBM System z is designed with built-in security capabilities to help protect your business and what IBM software is used to achieve this purpose.

Traditionally, when we think of security, we often think of home security—keeping the doors closed and locked, controlling access by limiting the number and distribution of keys, installing burglar alarms to detect physical intrusion, and installing smoke and carbon monoxide alarms to detect intrusion by other harmful substances. In many ways, IT security works in a similar fashion. You need systems that are designed to control access to the system, to detect and prevent intrusion into the system by unauthorized users, and to protect the system from corruption by unauthorized programs and viruses. In other words, you need to close and lock the doors and install a rigid and comprehensive set of fences and alarms to help protect against various types of intrusion.

It is a very complex environment that tries to interface into the z/OS and we must also deploy measures to protect communications to and from z/OS. IBM software on z/OS has this capability. We cover encryption, the use of certificates, key handling, and other measures such as auditing in this publication.

This chapter provides a brief overview of z/OS basic security and the additional security services under z/OS. z/OS security services comprise various security-related products, which are broadly grouped into three elements with some additional products, which we explain in detail in the following chapters. See Figure 1-1 on page 2 for z/OS basic security facilities.

## 1.1 z/OS basic security facilities

- Integrity
  - System authorization Facility (SAF)
  - Program property table (PPT)
  - Authorized program facility (APF)
  - Authorized programs
  
- Auditing
  - Logs (hardcopy, system)
  - Generalized trace facility
  - System management facility (SMF)

Figure 1-1 z/OS basic security facilities

The z/OS operating system is designed, implemented, and maintained to protect itself against unauthorized access, and thus security controls that are specified for that system cannot be compromised. Thus, there is no way for any *unauthorized* program, using any system interface, defined or undefined to:

- ▶ Bypass store or fetch protection
- ▶ Bypass the operating system password, VSAM password, or z/OS Security Server Resource Access Control Facility (RACF) checking
- ▶ Obtain control in an authorized state

### System authorization facility

The system authorization facility (SAF) is part of the operating system. SAF is available as part of the operating system. In addition, most installations also deploy an external security manager (ESM), such as the IBM z/OS Security Server. The different resource managers contact SAF. If an additional security product is installed, SAF routes the questions using the *SAF router* to the security product and routes the answer back to the resource manager. Thus, SAF builds the interface between the resource managers and the security product. The final decision, whether access will be granted, is made by the resource manager, not by SAF or the security product. See also Chapter 2, “System Authorization Facility” on page 9.

### Program property table

The program properties table (PPT) contains a list of programs that require special attributes. Among other things, the special attributes specify whether the programs can or cannot *bypass security protection* (password protection and RACF) and whether they run in a *system key*.

Programs with the NOPASS parameter are able to bypass password protection for password protected data sets and, thus, also bypass all RACF protection for RACF protected resources. Therefore, it can read any data set on the system.



The system key parameter indicates whether the program is authorized to run in a system key (system keys 0 - 7) and is thus able to bypass system security controls. Such a program can have access to fetch protected system data.

**Important:** You need to verify that only those programs that are authorized to bypass password protection are, in fact, able to do so. Such programs are normally communication and database control programs or other system control programs. You can also verify that only those programs that need to run in a system key are authorized to do so.

## Authorized program facility

Authorized program facility (APF) is a feature that allows system and user programs to use sensitive system functions. From a security and integrity perspective, the program must be authorized by the APF before it can use restricted functions such as supervisor calls (SVCs).

To authorize a program, the following steps are required:

1. The program load module must be marked as *authorized* by the binder or have the APF-authorized extended attribute bit set if the program resides in a UNIX System Services file system.
2. If loaded from a load module library the load library must be flagged as *authorized*, that is, this load module library must be in the APF list.
3. When the program is fetched, no non-authorized library can be part of the JOBLIB or STEPLIB concatenation.

## Authorized programs

Many system functions are sensitive (for example restricted SVCs). Therefore, these sensitive functions can be used only by authorized programs. An authorized program can virtually do anything it wants. You could consider it an extension of the operating system. Therefore, care is needed; such authorization must be given in a diligent manner and be monitored.

A program is *authorized* if one of the conditions is true:

- ▶ Program runs in supervisor state (bit 15 in PSW=0).
- ▶ Program runs in system protection key (bits 8-11 in PSW contains key 0-7).
- ▶ Program runs as part of an authorized job-step task (JSCBAUTH=1). This task is set if the initial program is marked AC=1 and if it is loaded from an APF-authorized library or from the link pack area (LPA). Or, has the APF-authorized extended attribute bit set if the program resides in a UNIX System Services file system.

## Auditing

z/OS has the following basic functions that provide information useful for auditing purposes:

- ▶ Logs (hardcopy and system)
- ▶ Generalized trace facility (GTF)
- ▶ System management facility (SMF)

An auditor needs to look closely at the security settings on the z/OS system. Most auditors have a good checklist to cover the areas where there is a need for vigilance and control. Some examples of this include:

- ▶ User entries in the PPT with bypass password protection (NOPASS) need to be verified and checked if still required.
- ▶ Determining whether the load libraries in the APF are all adequately protected.

An IBM product that provides this type of auditing capability is IBM Security zSecure™ Audit. It provides an audit solution for the z/OS operating system. It covers these areas:

- ▶ Identifies security weaknesses.
- ▶ Automates audit and compliance reporting.
- ▶ Tracks and monitors baseline and library changes.

Figure 1-2 shows z/OS Security Server components.

## 1.2 z/OS Security Server components



Figure 1-2 z/OS Security Server components

### Resource Access Control Facility

The Resource Access Control Facility (RACF) is a component of the z/OS Security Server. The z/OS Security Server is an optionally priced feature of z/OS that allows an installation to control access to protected resources and perform other security-related functions.

RACF helps meet your needs for security by providing the following abilities:

- ▶ Identify and verify users
- ▶ Authorize users to access the protected resources
- ▶ Control the means of access to resources
- ▶ Log and report attempts to access protected resources
- ▶ Administer security to meet an installation's security goals

RACF provides these functions when the installation defines the users and the resources to be protected.

In addition, RACF has the following capabilities:

- ▶ Act as a certificate authority
- ▶ Provide the environment to create, supply, and store digital certificates

Figure 1-3 shows Integrated Security Services components.

## 1.3 Integrated Security Services

- Network authentication service (KERBEROS)
- Enterprise identity mapping (EIM)
  - z/OS identity cache
- Open cryptographic enhanced plug-ins (OCEP)

Figure 1-3 Integrated Security Services components

Integrated Security Services is supplied as a base element of the z/OS operating system. It is not a single product but consists of the components described in the remainder of this section.

### Network Authentication Service

This is a service where it is assumed that the physical network cannot be trusted and provides an authentication service to the client and the application server that the client is trying to reach. It provides a network authentication service without having to send a user ID and password.

Network Authentication Service for z/OS performs authentication as a trusted third-party authentication service by using conventional shared secret-key cryptography. Network Authentication Service provides a means of verifying the identities of principals without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts on the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will.

Network Authentication Service for z/OS provides Kerberos security services without requiring that you purchase or use a middleware product such as Distributed Computing Environment (DCE). These services include native Kerberos application programming interface (API) functions, as well as the Generic Security Service application programming interface (GSS-API) functions.

Network Authentication Service for z/OS supports the following encryption types:

- ▶ 56-bit DES, referred to specifically as DES
- ▶ 56-bit DES with key derivation, referred to specifically as DESD
- ▶ 168-bit DES, referred to specifically as DES3
- ▶ 128-bit AES, referred to specifically as AES128
- ▶ 256-bit AES, referred to specifically as AES256

However, due to US government export regulations, DES3, AES128, and AES256 encryptions might not be available for user data encryption.

### Enterprise Identity Mapping

Enterprise Identity Mapping (EIM) is an architecture for describing the relationships between individuals or entities within an enterprise and the many identities that represent them in the enterprise. It also provides a set of APIs that allow applications to make inquiries about these

relationships. And it allows you to map a user's identity on one system to the user's identity on another system. See 4.17, "Overview of EIM" on page 194 for more information.

### Open Cryptographic Enhanced Plug-ins

Open Cryptographic Enhanced Plug-ins (OCEPs) provide an application interface for managing server certificates and also helps protect server private keys in a uniform and secure way. Applications that comply with Common Data Security Architecture (CDSA) standard interfaces can use OCEP. Open Cryptographic Services Facility (OCSF), which is a part of z/OS Cryptographic Services, provides these interfaces. Application developers and independent software vendors using OCEP can find it easier to develop and port applications to the z/OS operating system. It helps customers apply consistent security rules to e-business applications that use digital certificates and helps protect server private keys.

Figure 1-4 shows Cryptographic Services.

#### Notes:

- ▶ The Distributed Computing Environment (DCE) Security Server component was removed from the z/OS operating system with z/OS V1R13. An IBM Redbooks publication entitled *DCE Replacement Strategies* SG24-6935 has been written to show how it can be replaced.
- ▶ The LDAP server component was removed from the z/OS operating system with z/OS V1R11. It was replaced by IBM Tivoli® Directory Server for z/OS.
- ▶ The Firewall Technologies component was removed from the system with z/OS V1R8.

## 1.4 Cryptographic Services

- Integrated cryptographic service facility (ICSF)
- Open cryptographic services facility (OCSF)
- Public key infrastructure (PKI) services
- System secure sockets layer (SSL)
- PKI services trust policy (PKITP)

Figure 1-4 1.4, "Cryptographic Services" on page 6

*Cryptography* is the transformation of data to conceal its meaning. In z/OS, the base element Cryptographic Services provides the following cryptographic functions:

- ▶ Data secrecy
- ▶ Data integrity
- ▶ Personal identification
- ▶ Digital signatures
- ▶ The management of cryptographic keys

This base element supports keys as long as 56 bits. Keys longer than 56 bits are supported by the optional feature z/OS Security Level 3.

## **Integrated Cryptographic Service Facility**

Integrated Cryptographic Service Facility (ICSF) provides secure, high-speed cryptographic services in the z/OS environment. It provides APIs to support the encryption and decryption of data using the cryptographic hardware in the IBM System z servers.

The application calls ICSF for a cryptographic function and provides the data to be processed along with the cryptographic key to be used.

ICSF drives the cryptographic operations at the coprocessors and transmits and receives the processed data and the encrypted application key. Access to ICSF callable services and application keys can be controlled by RACF profiles.

## **Open Cryptographic Services Facility**

Open Cryptographic Services Facility (OCSF) is the z/OS implementation of Common Data Security Architecture (CDSA) API from Intel. OCSF actually uses ICSF to get access to the cryptographic hardware coprocessor.

Thus, it provides a means for applications to directly access security services through the ICSF security application programming interface or indirectly access security services via layered security services and tools implemented over the OCSF API.

## **Public Key Infrastructure services**

Digital certificates, in widespread use today, are becoming increasingly important as a means of helping to secure transactions on the Internet. As such, digital certificates add capabilities far superior to mere password protection. Public key infrastructure (PKI) services provides a trusted infrastructure that can manage and support the use of digital certificates. PKI services are provided as part of z/OS, so you can act as your own certificate authority (CA). As a CA, you have the power to create, approve or reject, and manage the lifecycle of digital certificates. Using PKI can represent significant savings to businesses currently purchasing digital certificates from third-party vendors.

## **System Secure Sockets Layer**

Secure Sockets Layer (SSL) consists of primarily two sets of APIs and a Certificate Management utility. The first set of APIs support the Secure Sockets Layer protocols (SSL V2.0, SSL 3.0, TLS V1.0, TLS V1.1, and TLS V1.2), which can be used by C/C++ applications to communicate securely across an open communications network.

The other set of APIs (Certificate Management) provides the ability to use functions other than the SSL protocols. These functions include the ability to create and manage key database files in a similar fashion to the SSL Certificate Management utility. They also include the ability to use certificates stored in a key database file, SAF key ring, or z/OS PKCS #11 token for purposes other than SSL and basic PKCS #7 message support. This provides application writers with a mechanism to communicate with another application through the PKCS #7 standard. This is a client-server protocol, with the client explicitly requesting an SSL communication.

## **PKI Services Trust Policy**

The PKI Services Trust Policy (PKITP) is an OCSF plug-in to perform certificate validation against an SAF key ring that contains a trusted CA or site certificate (called an *anchor certificate*) or a virtual key ring of either CERTAUTH or SITE certificates.

Server applications running on z/OS can use this function to verify certificates that other network entities (for example, users and other servers) present. PKI Services or other certificate authorities might have issued these certificates.

Figure 1-5 shows what products are supplied with z/OS Security Level 3.

## 1.5 Security Level 3

- z/OS Security Level 3
  - IBM Tivoli directory server for z/OS security level 3
  - OCSF security level 3
  - Network authentication service level 3
  - System SSL security level 3

*Figure 1-5 What products are supplied with z/OS Security Level 3*

This comprises a number of unpriced z/OS features, each having their own SMP/E function modification identifier (FMID). It permits the z/OS security services to conduct encryption with keys greater than 56 bits. However, its export is subject to US export regulations.

Figure 1-6 shows the IBM Tivoli Directory Server for the z/OS component.

## 1.6 IBM Tivoli Directory Server for z/OS

- Lightweight Directory Access Protocol (LDAP) server

*Figure 1-6 IBM Tivoli Directory Server for the z/OS component*

### **LDAP server**

The z/OS Lightweight Directory Access Protocol (LDAP) server, part of the IBM Tivoli Directory Server for z/OS product, is based on a client/server model that provides client access to an LDAP server. An LDAP directory provides an easy way to maintain directory information in a central location for storage, update, retrieval, and exchange.



# System Authorization Facility

When a request for a security service within z/OS is made, it is passed through the System Authorization Facility (SAF). This is the facility that provides the interface between system services and the external security manager (ESM) installed on the system. This is an established process, as SAF routes requests for authentication, resource access checking, and other security-related processes to the ESM through control points within the system services programs.

SAF supports the use of control points across products and across systems. Applications and system components call these control points in order to interface with the ESM. Security on z/OS is therefore centralized on SAF and the installed ESM. z/OS does not contain an ESM, although there are several software products that act as an ESM. In this book, we only discuss the ESM that is available from IBM for z/OS called Resource Access Control Facility (RACF), which itself is part of the IBM security server.

When there is no ESM installed, SAF creates the security constructs needed by system services. However, in reality it would be mandatory to have an ESM installed to meet today's exacting security requirements.

SAF provides an installation with centralized control over system security processing by using a system service called the *SAF router*. The SAF router provides a focal point and a common system interface for all products providing resource control.

**SAF router:** A service that provides a focal point for all resource control.

External security managers provide tables to SAF, which directs specific calls for security functions to specific routines within the ESM. The use of these tables allows z/OS to provide support for ESMs, thus giving the installation the flexibility to determine which ESM to use.

Figure 2-1 on page 10 highlights the major discussion points for SAF.

**Note:** SAF and the SAF router are present on all z/OS systems, regardless of whether an ESM is installed or not.

## 2.1 SAF overview

- What is SAF
- How does it work
- Auditing

Figure 2-1 SAF major points

### What is SAF?

It is a z/OS component supplied within the base of z/OS and is used by resource managers within z/OS to determine if an access is to be permitted. The System Authorization Facility can function on its own. It does not require another product as a prerequisite. However, overall security of z/OS is greatly enhanced when SAF works with another component to resolve these security questions. SAF rarely works on its own.

### How does it work?

There are execution points within any program code that manages security where decisions must be made. This is the point in the program logic where a determination must be made if the proposed action is to be permitted or allowed within this z/OS operating system.

The z/OS operating system refers to these points as control points. Examples include:

- ▶ Determining the validity of identity credentials
- ▶ Determining the authority of an identity to access a known resource
- ▶ Determining the nature of auditing actions

Figure 2-2 shows the security flow when a user is authenticated and validated.

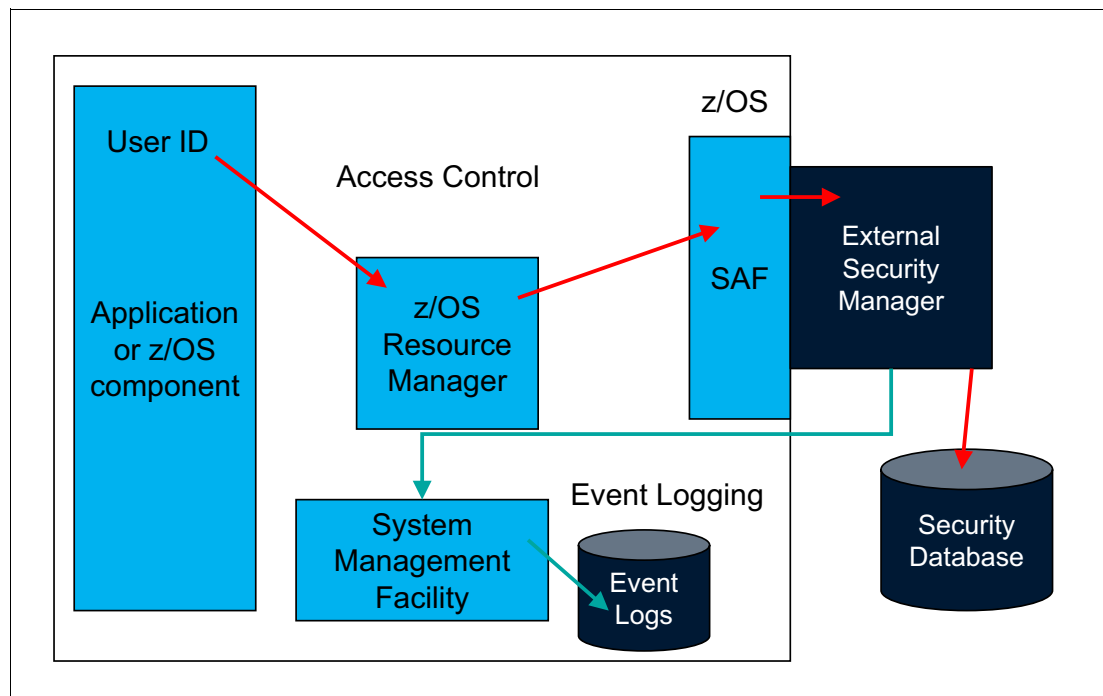


Figure 2-2 Security flow



The program logic at these decision points uses a z/OS operating system component known as the *System Authorization Facility* (SAF). This component is used by resource managers in making decisions regarding access to a resource.

Control points pass control to the SAF Router using the RACROUTE service (the name of this service shows its heritage in the early releases of RACF). SAF usually passes control to the z/OS feature, which is IBM's ESM and is called *Resource Access Control Facility* (RACF). Alternatively, SAF can be configured to pass control to another external security manager. There are alternate independent software vendor software offerings available that can be used in place of RACF. RACF and each of these other offerings are known as *external security manager*.

Figure 2-2 on page 10 shows the control flow when a user is authenticated and validated. It starts with the application (or z/OS component) communicating with a z/OS resource manager (for example, IBM IMS™), which makes a request to SAF via the RACROUTE service to verify a user ID. SAF passes this request to its ESM, which consults its database and passes the results back via the SAF to the application. You will see the logging of this activity via the System Management Facility (SMF) log records, which creates an audit trail.

## Auditing

We audit to record what security-related activities are occurring. A common example is when a user is authenticated to a system (they are logging on). A requirement may exist to log all users entering the system. In addition, as this user interacts within the system we may want to know what resources were accessed or more importantly what accesses failed.

We must also consider tracking when commands on the system are issued to change security information. It would be a serious flaw if you could not identify who altered security to permit access.

A feature of z/OS is the component called *System Management Facility* (SMF). This feature operates in a similar manner to the SAF resource managers. There are control points with the z/OS operating system where SMF records are written. In some cases, it will not be possible to bypass the controls over writing of these records.

In these examples here, we would see the z/OS operating system using what commonly is called the *SAF router*, but technically we say that the z/OS system service is using the RACROUTE REQUEST=AUDIT service to perform logging.

Figure 2-3 on page 12 shows detailed actions using SAF.

**Note:** In any z/OS operating system instance that has serious security requirements, the SMF security records are all selected to be written and the security messages are not suppressed.

## 2.2 SAF in detail

- SAF router
- Resource manager
- Validating identity credentials

Figure 2-3 Detailed actions using SAF

### SAF router

How do these programs perform system services access SAF from their control points? They use an authorized programming interface (API) to make this call. The actual program code used depends on the programming language used. But typically in this interface at the lowest level, it uses code that is created by the assembler macro called *RACROUTE*. Here we use the term RACROUTE service, but its programming implementation may well use another name as we see shortly.

SAF is accessed through the RACROUTE service. RACROUTE provides the services to authenticate a user ID, interrogate access permissions, perform security event logging, and obtain a security context for address spaces and tasks running on the system. Regardless of the ESM installed, applications and system services use the RACROUTE service.

The RACROUTE service in z/OS removes the need for the application requesting security services to understand the underlying system security infrastructure implemented by the installation. The application does not need to know which ESM is installed, or indeed if one is present at all.

The SAF router uses the routing table to associate the correct ESM programs with the related RACROUTE call, as illustrated in Figure 2-4 on page 13.

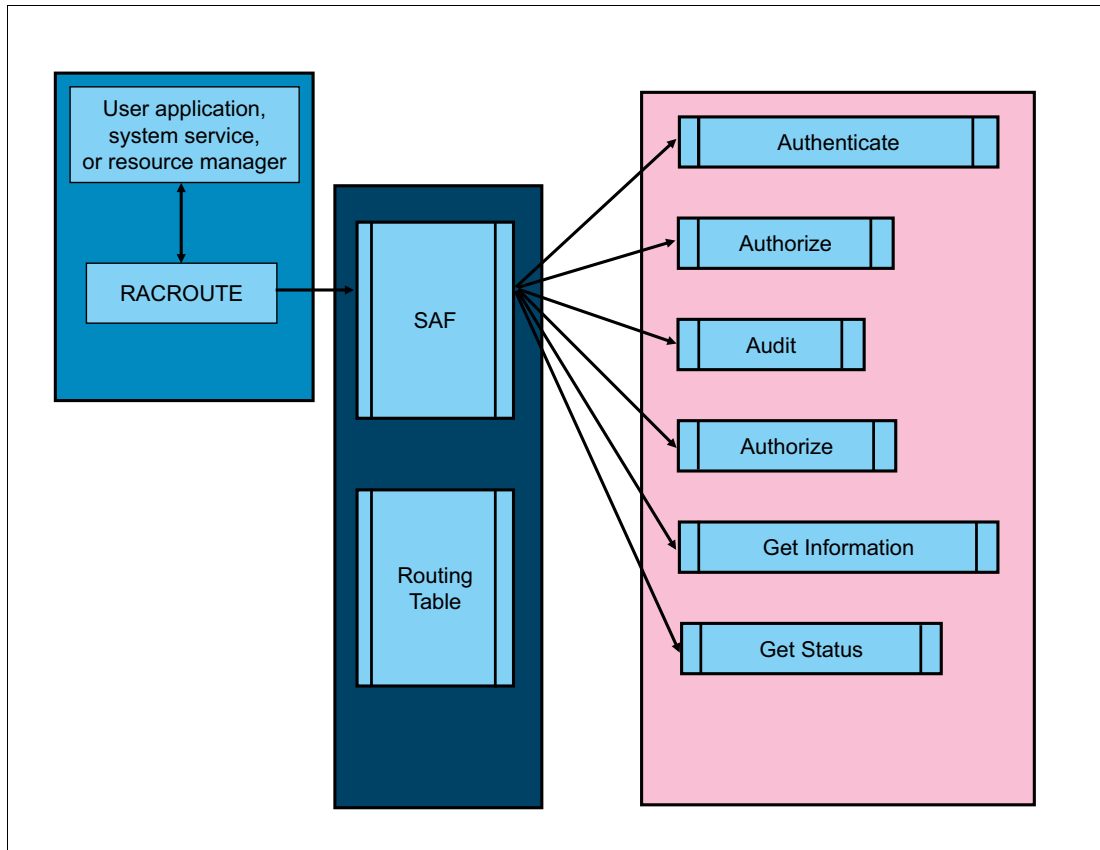


Figure 2-4 SAF router overview

Usually, when a user tries to access your system, whether through UNIX shell applications, FTP, a web page, some other network application, or even through TN3270 and TSO, authentication all uses a RACROUTE service. Understanding security on z/OS means understanding how programs interface with the external security manager. Independent of the ESM installed, applications need to initiate a RACROUTE service call or have one initiated on their behalf in order to gather the information needed for subsequent authorization checks.

Here we examine the mechanics of an authorization request by looking at the RACROUTE macro. RACROUTE, being the main access point into SAF, has many variants known as *request types*. Some sample request types are listed and described in Table 2-1, which gives an idea what requests are being made in the RACROUTE service.

Table 2-1 Some functions performed by RACROUTE

Request	Function
Audit	Record events in SMF type 80 records, and issue messages to the network security administrator.
Auth	Check a user's authority to access a resource.
Define	Define, modify, or rename a resource profile.
DirAuth	Compare two security labels.
Extract	Retrieve or replace certain specified fields from a protection profile.
FastAuth	Verify access to resources whose protection profiles have been brought into main storage by the RACROUTE REQUEST=LIST service.

A RACROUTE service call is initiated within the resource managers on z/OS. A typical application would not code a RACROUTE directly. For example, applications written in C/C++ use `pthread_security_np()` or `__passwd()` calls instead.

The user requires a security context to access z/OS resources. The WebSphere Application Server issues a `pthread_security_np()` to create the thread level security for the user. The `pthread_security_np()` resolves, deep down in the dark recesses of z/OS as a RACROUTE REQUEST=VERIFY, ENVIR=CREATE service call. Example 2-1 shows an authentication check using the RACROUTE REQUEST=VERIFY service call. This example only shows what the source code would look like in the assembler. It would be different using other programming languages such as C++, or even Java; using the binder, the executable form of this example would be available to the application that needs to establish a security context.

*Example 2-1 Example in source code of using the RACROUTE service call to verify a user ID*

---

```
label RACROUTE REQUEST=VERIFY,ENVIR=CREATE,USERID=USERDATA,PASSWRD=USRPASS,RELEASE=2.2,MF=S
```

This example assumes that USERDATA and USRPASS have been defined.

---

## Resource managers

Resource managers might be independent sections of program code or they may be embedded into other modules and routines. It is usual for resource managers to reside in authorized code. (However, in certain situations it is possible for a resource manager to perform satisfactorily without the code being authorized.) The role of a resource manager is to control access to a resource. So, for example, in the case of an attempt to access a data set, this is performed in the OPEN SVC routine in z/OS.

Each time that a data set is accessed for the purposes of reading or writing, the OPEN SVC is used first. Only after this OPEN SVC has completed without error can a program perform read and write operations to the data set. So this OPEN SVC is started by the program to prepare the data set for access, and also (via its resource manager) to check that the user running the program has the necessary access to the data set.

When the resource manager performs this check, it uses the RACROUTE REQUEST=AUTH or RACROUTE REQUEST=FASTAUTH service. If an external security manager is present, this results in a check being made that is based on the following types of questions:

- ▶ Who is attempting the access?
- ▶ What is the class of the resource?
- ▶ What is the name of the resource?
- ▶ What type of access is being requested (for example, READ or WRITE)?

When RACROUTE passes control back to the resource manager, the resource manager can examine the return code and reason code values and can then decide based on those codes.

It should be emphasized here that it is the resource manager that makes decisions about access using the information extracted from the external security manager.

## Validating identity credentials

Let us examine an example where SAF is used to authenticate a user.

Any program that performs the role of validating identity credentials should not be capable of being subverted in any way. Therefore, for the protection of such programs they can hold data in a storage key other than key 8. This requires that they be authorized either by running in supervisor state, being specified to run in a non-user key (that is, not key 8), or running with APF authorization. In practice most of these applications run with APF authorization and then move into a system key or supervisor state as necessary.

There are several z/OS components that perform this type of operation, and several major software applications that also perform this type of operation, such as:

- ▶ TSO logon (a part of z/OS)
- ▶ Batch job verification (performed by JES, which is a part of z/OS)
- ▶ APPC initiators (APPC is a component of z/OS)
- ▶ IBM Tivoli NetView®
- ▶ IBM CICS®
- ▶ IMS
- ▶ IBM WebSphere® Application Server

It should be apparent that these software components must be trusted components in a similar manner as the z/OS operating system itself.

The software performing this process of identity verification uses a system service called *RACROUTE REQUEST=VERIFY*. It passes to these RACROUTE service items, such as:

- ▶ Name of the user ID whose identity is being established
- ▶ Associated terminal identification or port of entry
- ▶ Identity verification credentials (for example, password, password phrase, PassTicket)

The application also has access to other information such as the time of day, the date, and other environmental information. After the call to the ESM (through SAF) the authenticating component uses the extracted data to either permit the authentication (such as permitting a logon) or to deny the access.





## IBM z/OS Security Server RACF

The operating system provides integrity. By using an external security manager, in this case Resource Access Control Facility (RACF), you can protect resources by defining which resources are protected and which groups of users or which individual users have access to the defined resources. The definitions are kept in the RACF database. A RACF administrator defines users, user groups, and resources together with rules for how these resources can be used. RACF is “invisible” for most users if a good security structure is put in place. Most companies have well-documented policies for information security. All RACF definitions need to be based on these policies.

RACF helps meet the needs for security by providing the following abilities:

- ▶ Identify and verify users
- ▶ Authorize users to access the protected resources
- ▶ Control the means of access to resources
- ▶ Log and report attempts to access protected resources
- ▶ Administer security to meet an installation's security goals

RACF provides these functions when the installation defines the users and the resources to be protected.

A specific RACF user, called the *security administrator*, has the responsibility to define users and resources to RACF. The security administrator also specifies the rules that RACF uses to control access to the resources.

The responsibility to implement the guidelines falls to the *system programmer*, who provides technical support for RACF. The system programmer installs RACF on the system and maintains the RACF database. This person oversees the programming aspects of system protection and provides technical input on the feasibility of the implementation plan. In addition, the *technical support person* can write and implement RACF installation exit routines to extend the security infrastructure. RACF retains information about the users, resources, and access authorities in *profiles* in the RACF database and refers to the profiles when deciding which users are permitted access to a protected system resource. The *auditor* monitors the security controls and checks that the security goals are met.

Figure 3-1 on page 18 defines RACF.

## 3.1 What is RACF?

### What is RACF?

- RACF, Resource Access Control Facility is an ESM product to implement and control the installation's security policies on z/OS systems.
- Access to protected resources is controlled by rules.
- Access to resources are logged and can easily be monitored by an Auditor
- Users, groups and resources together with access rules are administrated by an Administrator

*Figure 3-1 What is RACF?*

### What is RACF?

RACF is an optional external security manager (ESM) software product that provides basic security to a z/OS system. Other security software products are also available. RACF as a component of z/OS Security Server is included as part of the base z/OS system but requires a separate license to be activated.

RACF provides the ability to implement the security policies that you choose on your system.

Figure 3-2 on page 19 shows RACF functions.

**Note:** Your system will not be secure by simply installing RACF. The quality of the system protection depends on the way that you use the RACF functions.



## 3.2 RACF functions

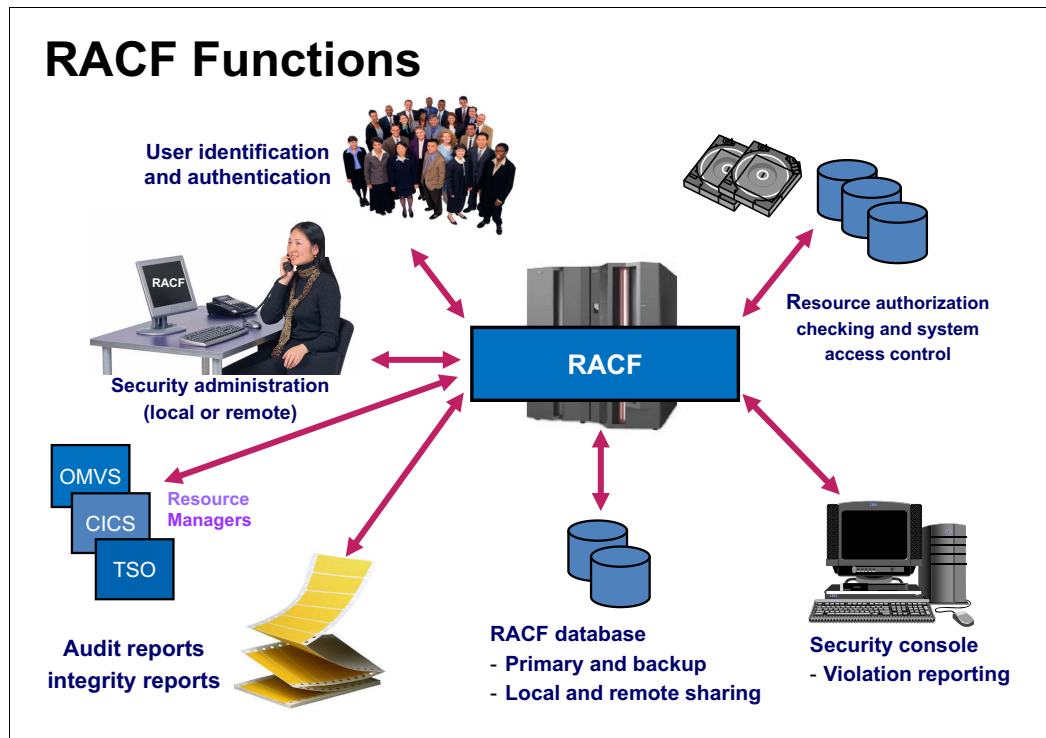


Figure 3-2 RACF functions

RACF protects resources by granting access only to authorized users of the protected resources. To enforce your security policy, RACF gives you the ability to accomplish the tasks described in this section.

### Identify and authenticate users

User authentication is validation of the user requesting access. The first step is to *identify* the person who is trying to gain access to the system, and the second is to *authenticate* that the user is really that person. The standard approach to RACF user identification is achieved by the use of a *user ID* and *password phrase* or *password* to perform user identification and authentication. Other options are available, such as digital certificates, use of a smart card, and the use of PassTickets.

RACF also has the concept of GROUPs, where users may be connected to a GROUP in order to gain access to the resource. This greatly simplifies the matter as the security administrator does not have to grant every individual user access to a particular resource by defining a rule for each user, but instead a simple connection to an appropriate group (which has access to the required resource) will suffice.

RACF will also permit identity propagation from other systems. Identity propagation is the capability whereby a non z/OS identity, a distributed identity, is propagated into the z/OS environment and is then used to provide credentials for authorization by being mapped to an existing RACF user ID and is available throughout the z/OS Sysplex for auditing and reporting.

## Resource authorization

Having identified and verified the user, RACF then controls interaction to the system resources. RACF must authorize the users who can access resources and also the way users can access them, which depends on the purpose of each user (for example, reading or updating). RACF can also authorize when a user can access resources, by either time or day.

## Log and report access to protected resources

RACF provides the ability to *log* information, such as an attempted access to a resource and to generate *reports* containing the information that allows identification of users who attempt to access resources. Following are the logging and reporting functions:

- ▶ **Logging:** RACF writes records to the system management facility (SMF) data set for unauthorized attempts to enter the system and optionally RACF writes records to SMF for authorized attempts. Other events can also be logged.
- ▶ **Reporting:** The SMF records can be analyzed by the *RACF Report Writer* or be processed by other reporting software such as IBM Security zSecure Audit.

**Important:** The *RACF Report Writer* was stabilized at the RACF 1.9.2 level. It cannot produce reports for new SMF records available beyond that release.

- ▶ **Sending Messages:** RACF sends messages “real time” to the *security console* and, if implemented, to RACF defined TSO users as well.

## Security administration

RACF can be administered either in a *centralized* or *decentralized* manner. In a centralized approach, the RACF administrator (user attribute SPECIAL) controls the access to all users, groups, and resources.

In a decentralized approach, RACF administration can be delegated to administrators only at a group level. These administrators have the group-SPECIAL attribute, which enables them to control access only to their group or to be more precise to their *scope* of the group. The scope of control of a group-level attribute percolates down through a group-ownership structure from group to subgroup to subgroup, and so on. Percolation is halted (and, therefore, the scope of control of the group-level attribute is ended) when a subgroup is owned by a user instead of a superior group.

Another way to implement decentralized administration is by use of *class authorization*. To do this, an administrator is authorized only for specific types of profiles, for example, for user profiles. In this case, the administrator can administer user IDs but cannot define how resources are protected or who should have access to resources.

## RACF database

RACF holds information about the users, groups, resources, and access authorities in profiles that are stored in the RACF database and refers to the profiles when deciding if users are permitted access to protected system resources. Applications can request RACF services. Most of these services can only be requested by authorized applications.

RACF processing uses the information from the database each time a RACF defined user enters a system and each time a user wants to access a RACF protected resource. Some of this information can be cached in storage.

You maintain the RACF database through commands, assembled macros, and utilities.

The RACF database is a non-VSAM, single extent data set that is made up of 4 KB blocks and must be cataloged. A very detailed explanation on the database layout can be found in the *Security Server RACF Diagnosis Guide*, GA32-0886.

RACF allows you to provide a *backup* database to which you can switch without a restart in case your *primary* RACF database fails. A backup RACF database reflects the contents of the primary database. After the installation has created the backup database, RACF can maintain it automatically. In addition, the RACF database can be shared between other z/OS and z/VM systems.

### **Interaction with major subsystems**

RACF does not operate in isolation, but interacts with these subsystems on z/OS:

- ▶ Storage Management Subsystem (SMS): RACF is used to protect and control the use of SMS classes, data sets, functions, options, and commands.
- ▶ z/OS UNIX: Security functions provided by RACF include user validation, file access checking, privileged user checking, and user limit checking.
- ▶ Time Sharing Option Extensions (TSO/E): In order for users to log on to TSO, they must have an entry in the SYS1.UADS data set or a TSO segment defined in their RACF user profile.
- ▶ Job Entry Subsystem (JES): The JES requests RACF services by issuing a RACROUTE request. The z/OS system authorization facility (SAF) handles the RACROUTE request. If RACF is installed, SAF passes the security information specified by JES on the RACROUTE request to RACF. RACF evaluates the security information and returns the results of that evaluation to JES. JES then enforces the results of the security check, such as permitting or denying access to a data set, or allowing a job to execute. JES here covers JES2 or JES3.
- ▶ IBM Tivoli Directory Server for z/OS (Lightweight Directory Access Protocol (LDAP)): The LDAP server can be configured to provide access to RACF and store application-specific information and also provide read/write access to RACF user, group, connection, and general resource profiles using the LDAP protocol. The server can also be used to manage RACF options that affect classes.

Figure 3-3 on page 22 shows the RACF primary Interactive System Productivity Facility (ISPF) panel.

## 3.3 RACF ISPF panel

```
RACF - SERVICES OPTION MENU
OPTION ===>

SELECT ONE OF THE FOLLOWING:

  1  DATA SET PROFILES
  2  GENERAL RESOURCE PROFILES
  3  GROUP PROFILES AND USER-TO-GROUP CONNECTIONS
  4  USER PROFILES AND YOUR OWN PASSWORD
  5  SYSTEM OPTIONS
  6  REMOTE SHARING FACILITY
  7  DIGITAL CERTIFICATES, KEY RINGS, AND TOKENS
 99  EXIT

      Licensed Materials - Property of IBM
      5650-ZOS Copyright IBM Corp. 1983, 2013
      All Rights Reserved - U.S. Government Users
      Restricted Rights, Use, Duplication or Disclosure
      restricted by GSA ADP Schedule Contract with IBM Corp.
```

Figure 3-3 RACF primary ISPF panel

### How to use RACF ISPF panels

If your installation has installed the RACF panels, you can use them to perform security tasks.

To access the RACF panels, enter the following command:

```
ISPF
```

The Interactive System Productivity Facility (ISPF) primary menu displays. From this menu, choose option R for RACF.

**Note:** Although this method is the usual way to access RACF panels, your installation might have this implemented through a different path.

The RACF panel interface is similar in use to all other ISPF panel options. Therefore, we do not go into detail here on to how to use it.

You can access help information for the RACF panels. Help panels exist for each individual panel. If you have a question about the information that you should provide on the panel, either press PF1 or type HELP on the command line. The help panels give more information about the terms on the panel and the information that you need to enter.

Figure 3-4 on page 23 shows RACF resource profiles.

## 3.4 RACF profiles

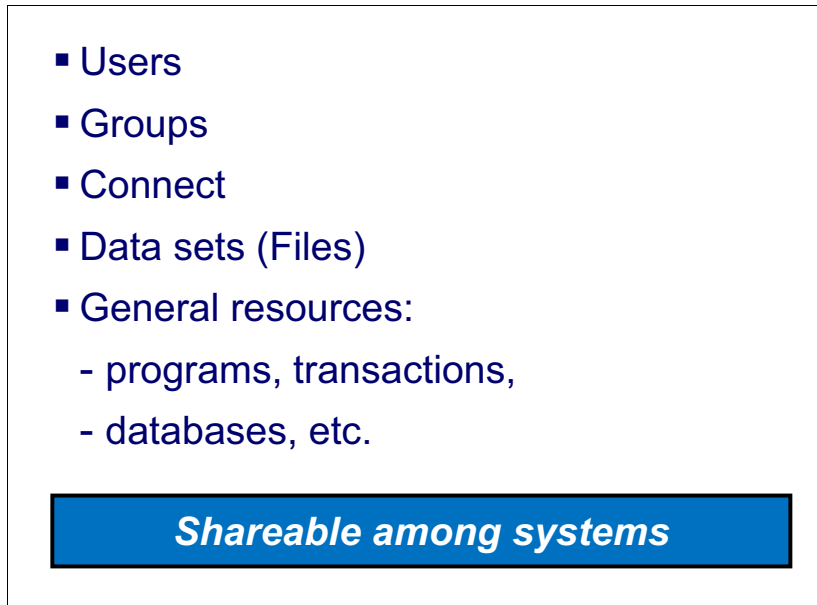


Figure 3-4 RACF resource profiles

RACF maintains information entries, called *profiles*, in the RACF database.

### User profile

When a new user is created, its profile is added to the RACF database. It contains a RACF segment but may also contain segments such as TSO, DFP, and OMVS. Each segment of a user profile consists of fields. When you define a user's profile (using the **ADDUSER** command) or change a user's profile (using the **ALTUSER** command), you can specify the information contained in each field of each segment of the profile.

### Group profile

When you define a group to RACF, you create a group profile in the RACF database. A group profile consists of *segments*: a base segment and optionally, CSDATA, DFP, OMVS, OVM, and TME segments. Each segment of a group profile consists of *fields*. When you define a group's profile (using the **ADDGROUP** command) or change a group's profile (using the **ALTGROUP** command), you can specify the information contained in each field of each segment.

### Connect profile

This is created as a by-product of creating other profile entries as it establishes linkages between RACF objects. For example, a connection profile is created when a user is connected to a group.

### RACF resource profiles

USER, GROUP, and CONNECT profiles are not resource profiles. RACF protected resources can be divided into two categories:

- ▶ Data sets
- ▶ General resources

These profiles are used to protect DASD and tape data sets and general resources, such as tape volumes and terminals:

- ▶ Data set profiles contain security information about DASD and tape data sets.
- ▶ General resource profiles contain security information about general resources.

Each RACF defined resource has a profile, though you can optionally use single profile to protect multiple resources.

RACF commands or the RACF ISPF panels can be used to create and modify general resource profiles.

RACF provides discrete, generic, and grouped resource profiles for both data sets and general resources, as follows:

**Discrete** Discrete profiles have a one-for-one relationship with a resource: one profile for each resource. Discrete profiles provide very specific levels of control. Use them for sensitive resources. They protect only the one identified data set that is on the specified volume or that spans specific volumes. For example, a single data set can be defined with a discrete profile to allow access by one user.

**Generic** Generic profiles have a one-for-many relationship. One profile controls access to one or more resources whose names contain patterns or character strings that RACF uses to associate them with each other. They contain a list of the authorized users and the access authority of each user. A single generic profile can protect many data sets that have a similar naming structure. For example, all data sets that have a high-level qualifier of SMITH and the characters DATA as a second-level qualifier can be controlled with one generic profile.

**Grouped** Another type of RACF profile is the grouped profile. There might be no way to associate the resources with a common access list based on patterns in the resource names. In this case, the many resource names can be associated with a single RACF profile by using a grouping profile that contains the names of the associated resources.

Some subsystems with high performance requirements, such as IMS and CICS, have the profiles resident in the subsystem address space. These subsystems can save main storage by using grouped profiles.

Figure 3-5 on page 25 shows a list of RACF commands.

## 3.5 RACF commands

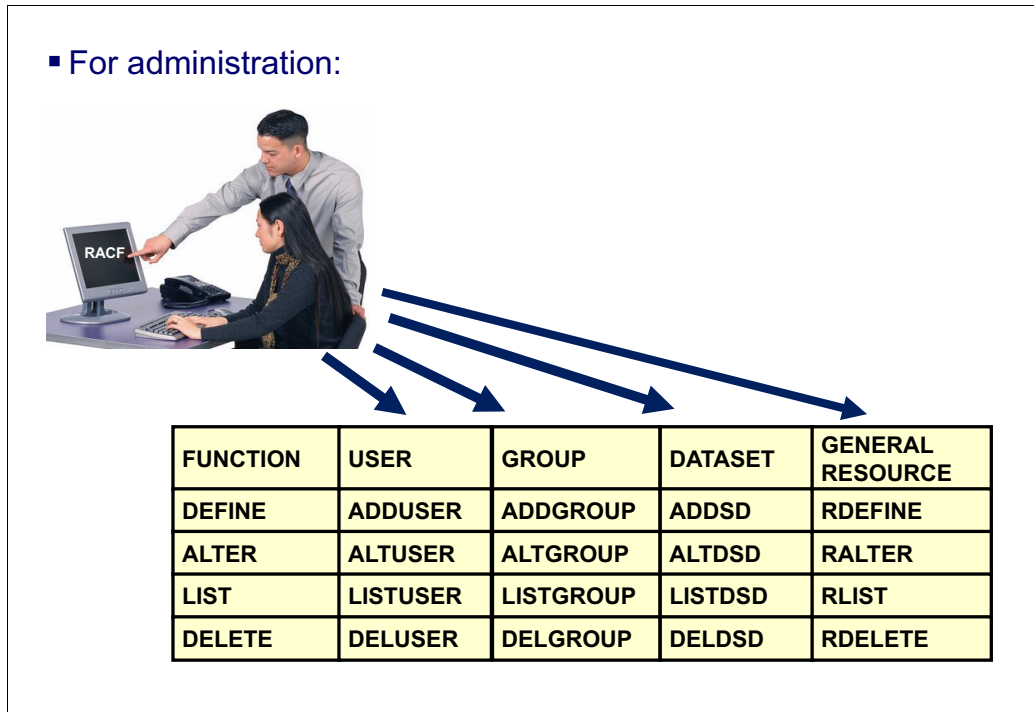


Figure 3-5 RACF commands

### Administration

For each type of entity in RACF, a set of commands is available to define, modify, list, and delete them.

There are several ways to enter RACF commands:

► RACF TSO commands

Easy and appropriate for ad hoc displays and update of user profiles and data set profiles, for example:

```
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
```

```
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(JANE) ACCESS(READ)
```

► RACF TSO commands in batch

Most appropriate for a set of displays that is run, unchanged, at regular intervals.

► RACF ISPF panels

Might be most appropriate for display of some of the more complex RACF general resource profiles. They are also very useful if you do not know the syntax for a particular command.

► Use of other software products

Such as IBM Security zSecure Admin product will facilitate the initiation of RACF commands.

In general, you must have authority for a RACF entry in order to display results from the command. A normal TSO user can display only the RACF data relevant to himself. A user with SPECIAL authority can display almost anything.

**Note:** We say *almost* because RACF has another authority named AUDITOR, who can uniquely display certain statistical data. A SPECIAL user can create AUDITOR authority, so the SPECIAL user remains the ultimate controller of RACF.

## Using RACF commands with TSO/E

You can enter RACF TSO commands from the ready prompt or by selecting Option 6 from the ISPF menu.

You can get online help for RACF commands. To get online help for a command, type:

```
HELP command-name
```

For example, to see online help for the **PERMIT** command, enter:

```
HELP PERMIT
```

To limit the information displayed, use the SYNTAX operand on the **HELP** command:

```
HELP command-name SYNTAX
```

For example, to see only the syntax of the **PERMIT** command, enter:

```
HELP PERMIT SYNTAX
```

General use RACF commands include:

<b>PASSWORD</b>	Change password/interval
<b>CONNECT</b>	Associate user with group
<b>REMOVE</b>	Disassociate user from group
<b>PERMIT</b>	Modify resource profile access list
<b>SEARCH</b>	Locate RACF information
<b>SETROPTS</b>	Set/modify RACF system options
<b>RVARY</b>	Switch RACF databases
<b>ADDUSER</b>	Create a new user
<b>LISTGRP</b>	List information on a group

You can also use abbreviations for most RACF commands:

- ▶ PW for PASSWORD
- ▶ CO for CONNECT
- ▶ RE for REMOVE
- ▶ PE for PERMIT
- ▶ SR for SEARCH
- ▶ SETR for SETROPTS
- ▶ RVARY has no abbreviation
- ▶ AU for ADDUSER
- ▶ LG for LISTGRP



You can use any TSO commands in a batch job, using the JCL for executing the TSO monitor in batch, as shown in Figure 3-6.

```
//BATCH JOB 1,P390,MSGCLASS=X
//TSOBAT01 EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUADS DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSTSIN DD *
LD DA('MARTIN.*') AUTHUSER
LU MARTIN
/*
```

Figure 3-6 JCL example of executing RACF commands in a batch job

Where the following command **LISTDSD** (LD) lists the generic profile MARTIN and its access list:

```
LD DA('MARTIN.*') AUTHUSER
```

And, the following command **LISTUSER** (LU) displays the basic RACF data for the user ID MARTIN:

```
LU MARTIN
```

Figure 3-7 shows the user authentication process.

### 3.6 User authentication

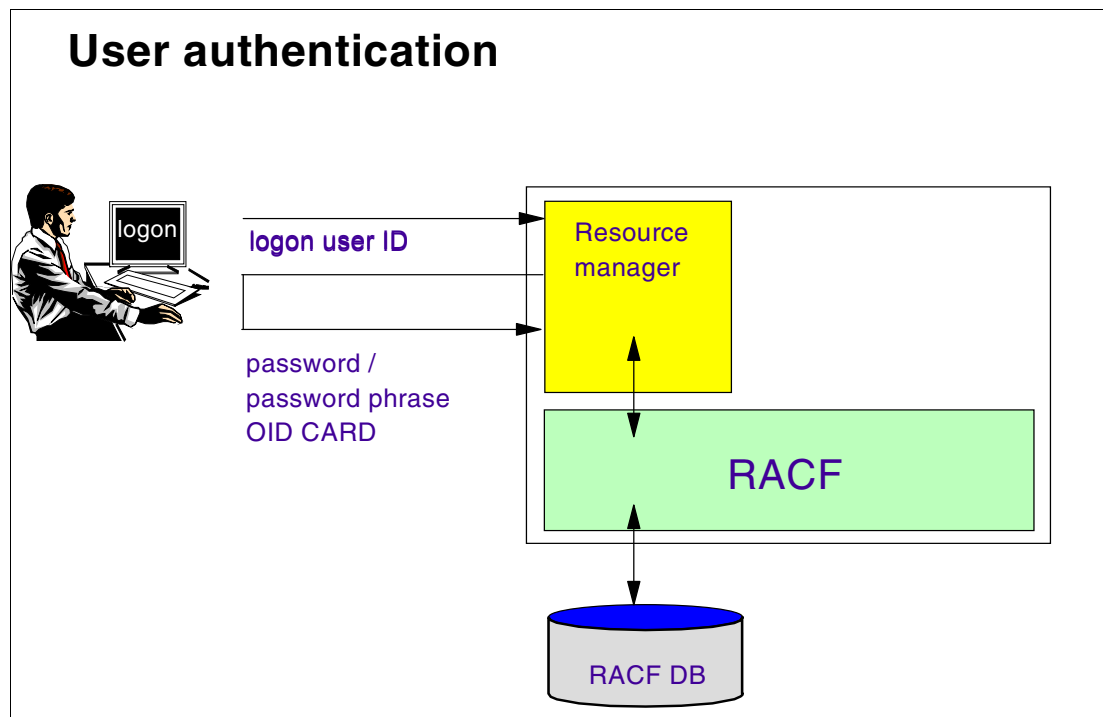


Figure 3-7 User authentication

RACF identifies and authenticates users accessing the system when the various system resource managers (such as TSO during a logon attempt) request it. RACF determines the following conditions:

- ▶ Whether the user is defined to RACF.
- ▶ If the user has supplied a valid password, password phrase, PassTicket, or operator identification card (OIDCARD) and belongs to a valid group. RACF has support for a password phrase that can be up to 100 characters long.
- ▶ If the user accesses a UNIX System Services resource, the user also must have a valid UNIX user identifier (UID) and UNIX group identifier (GID) (if this is not provided by a default user and group ID).
- ▶ Whether the user ID is in REVOKE status, which prevents a RACF defined user from entering the system at all or entering the system with certain groups.
- ▶ If the user can use the system on this day and at this time of the day (an installation can impose restrictions).
- ▶ If the user is authorized to access the terminal (which can also include day and time restrictions for accessing that terminal).
- ▶ If the user is authorized to access the application.

Beginning with z/OS Version 1 Release 11, RACF accepts information about the identities of distributed users from authorized applications, this is called *identity propagation*.

Figure 3-8 shows the resource managers.

### 3.7 Resource managers

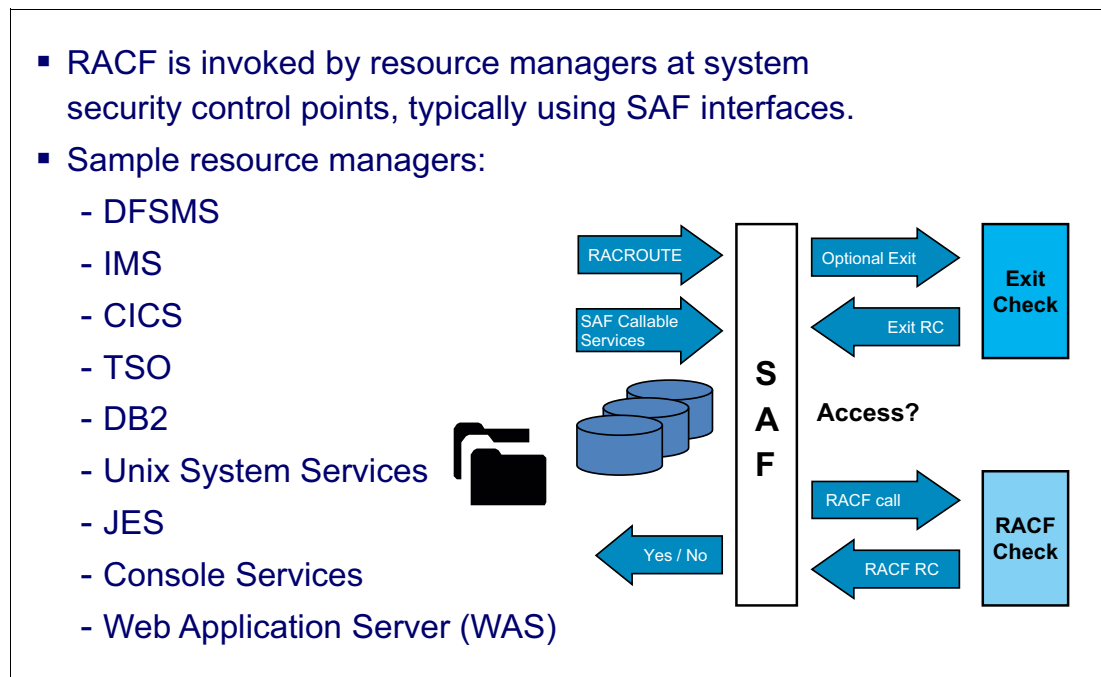


Figure 3-8 Resource managers

After the user has been authenticated, RACF controls access to resources. Before the user can access a protected resource, RACF ensures that the user is authorized to use the resource in the intended way (read, update, day, time, and so forth).

So how would a user be allowed to gain access to resources on a z/OS operating system? A typical check to authorize a user for access to a particular resource would follow these steps:

- ▶ A user is identified and verified to the RACF protected system.
- ▶ A user wants to modify or access an existing RACF protected resource. This will typically be in their interaction with a subsystem we call a *resource manager*. For example, they are interacting with IBM DB2®.
- ▶ The system resource manager (such as DB2) processes the request.
- ▶ The resource manager issues a RACROUTE request, which travels via SAF into RACF to ask if the user is permitted to perform their action.

**Note:** The resource manager is responsible for initiation of the authorization check.

- ▶ RACF checks one profile to verify that the user can access the resource and to determine whether the user has the required authorization to modify the contents.
- ▶ RACF returns via SAF the results of this check to the resource manager.
- ▶ The resource manager, based on what RACF indicates, makes the decision to either grant or deny the request. So, you can see that the resource managers that make the RACROUTE request to SAF must be deemed trusted in order to perform their functions.

Figure 3-9 on page 30 shows the RACF classes.

## 3.8 RACF classes

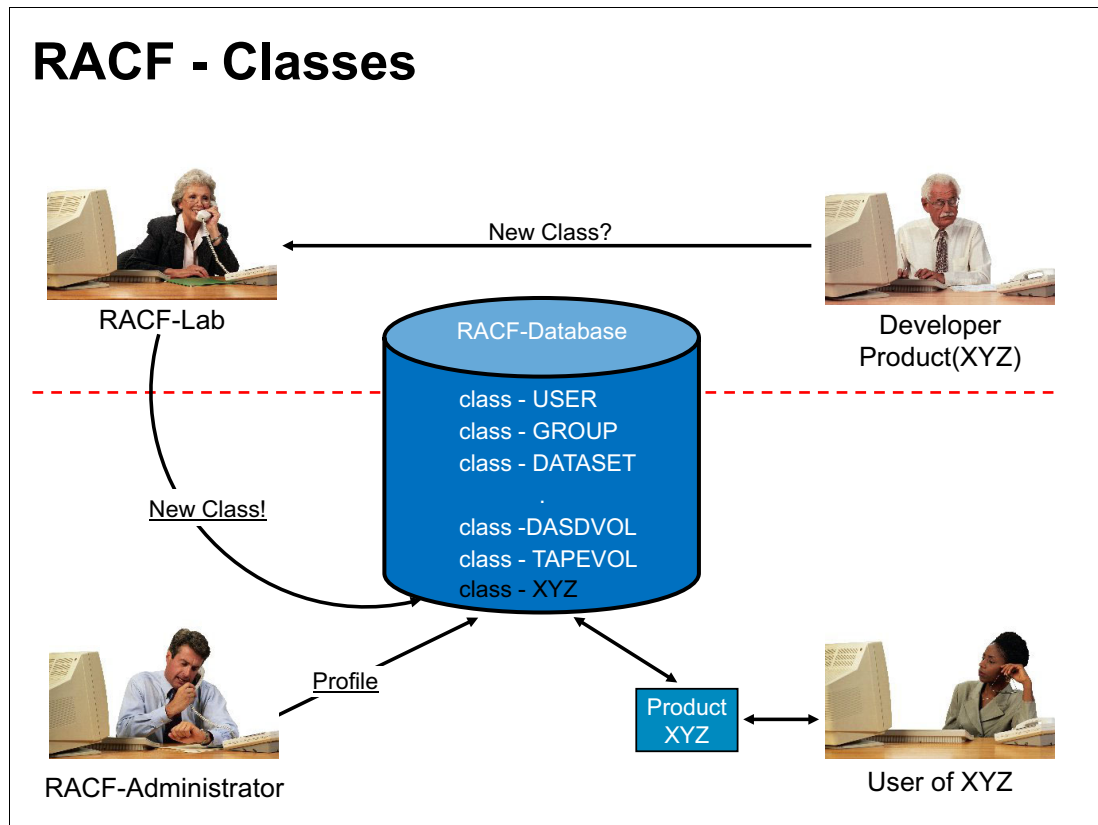


Figure 3-9 3.8, "RACF classes" on page 30

RACF stores information about users, groups, and resources in the RACF database. The information is normally kept in storage to enhance performance. The drawback is that this data must be refreshed when data is changed; updating the in-storage copy is done through the **SETROPTS** command.

### RACF: Administrator

To protect resources, the RACF Administrator needs to know in which classes a resource manager keeps the RACF information. This information is normally documented in the z/OS Security Server RACF reference manuals.

The RACF administrator defines user profiles in the RACF class *USER*, group profiles in the class *GROUP*, resource profiles for data sets in the class *DATASET*, and resource profiles for tapes in the class *TAPEVOL*.

It is possible to define additional classes. You can do this by modifying the dynamic *class descriptor table* (CDT) and then activating the updated table. You can find the general resource classes in the supplied CDT. It can be found in Appendix A of *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

Figure 3-10 on page 31 shows the security administration with RACF.

**Note:** The class descriptor table can be updated dynamically.

## 3.9 Security administration with RACF

- Set RACF system options
- Define users
- Define groups
- Define resource profiles
  - data sets
  - general resources
- ISPF Panels, RACF commands,
- TSO commands and optionally additional product like IBM Security zSecure Admin

Figure 3-10 3.9, "Security administration with RACF" on page 31

The administrator is a user with the SPECIAL user attribute. As a security administrator, you are the focal point for planning security and maintaining security at your installation. You need to:

- ▶ Determine which RACF functions to use and how these functions are to be used
- ▶ Identify the level of RACF protection
- ▶ Identify what resources RACF is to protect
- ▶ Identify administrative structures (centralized or decentralized)
- ▶ Decide on naming conventions (for example for groups and user IDs)

A RACF security administrator performs the tasks that we describe in this section.

### Define RACF system options

The key factor is to understand what RACF functions to use and to use these functions to achieve your security goals. Questions for the security administrator to consider and then set the system-wide options accordingly include:

- ▶ Naming conventions?
- ▶ Data set protection for all data sets?
- ▶ Resource protection for which classes?
- ▶ Group structure?
- ▶ RACF tailoring?
- ▶ Transparency?
- ▶ Recovery?
- ▶ Violation detection?
- ▶ Subsystems?
- ▶ Networks?
- ▶ Data sharing?

### Define user IDs and assign attributes

Individual accountability should be one of your installation's prime security objectives. RACF offers you the ability to assign each user a unique identifier. (Of course, whether you establish this degree of accountability in all cases is an installation decision.) A RACF user is identified by an alphanumeric *user ID* that RACF associates with the user. The maximum length of a

user ID from the RACF point of view is eight characters, but the maximum length for TSO is seven characters. Some users have particular tasks and, therefore, have *attributes* assigned. Some examples of attributes include:

- ▶ SPECIAL for a system-wide security administrator
- ▶ AUDITOR for a person who has overall responsibility to monitor the security guidelines
- ▶ OPERATIONS to give the user the system-wide OPERATIONS access

The information about the user is stored in the *user profile*.

When defining a user it is mandatory to name the *default group* of the user. Each RACF defined user belongs at least to his default group, but can be a member of multiple groups. Furthermore it is necessary to have an *owner* of the user profile. Normally the default group is chosen as owner.

## Define groups

A user is connected to one or more groups. The information about the group is stored in the *group profile*. A RACF group normally contains a number of users who share common access requirements. It is important to consider the basic purpose of a group, for example whether it is an administrative group, a holding group, a data control group, a functional group, or a user group? Beyond this consideration, it is necessary to specify the owner of the group.

The SPECIAL, AUDITOR, and OPERATION attributes are also applicable at the group level. The authority of the group-SPECIAL, group-AUDITOR, and group-OPERATIONS users is limited to the *resources* that are within the scope of the group.

**Important:** The *owner* in RACF relates to the profile. The owner of a profile can update the profile.

## Define RACF resource profiles

Appropriate protection of resources is an important goal that the security administrator must achieve. RACF maintains these information entries in *resource profiles* in the RACF database. It uses them to protect DASD, tape data sets, and general resources, such as transactions, programs, or spool output. RACF uses two kinds of resource profiles:

- ▶ Data set profiles contain security information about DASD and tape data sets.
- ▶ General resource profiles contain security information about general resources.

**Note:** In most cases, multiple resources are protected with a single profile, referred to as *generic* profiles.

## ISPF panels and commands

You can define most RACF functions using RACF ISPF panels. This interface is very useful for definitions or updates of a few entries. If you need to change many entries, TSO commands, maybe in combination with REXX programs, are often a better alternative.

The RACF operator commands allow you to perform functions in the RACF subsystem. You can enter these commands from an operator console. These commands allow a z/OS operator to perform certain RACF operations in the RACF subsystem. The RACF subsystem prefix in front of the command identifies the RACF subsystem as the processing environment.

Many RACF commands can be entered using TSO/E. RACF commands can also be issued from security-related software products such as IBM Security zSecure Admin. This is a powerful tool to perform small and large-scale RACF commands.

Figure 3-11 on page 33 shows RACF user identification and verification.

## 3.10 RACF user identification and verification

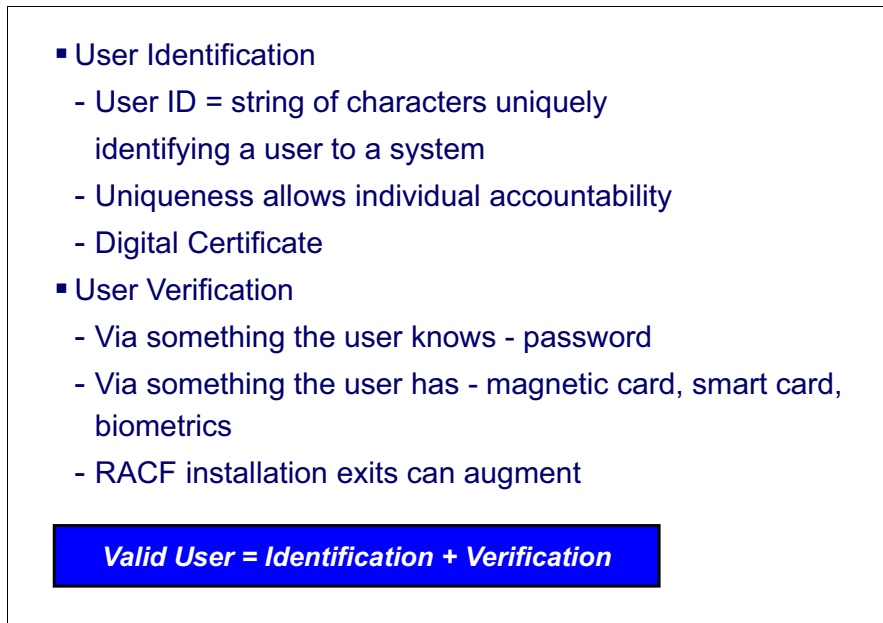


Figure 3-11 RACF user identification and verification

All users must be defined to RACF directly or indirectly in the case of identity propagation, where a RACF user ID is mapped to one or more distributed user identities. Users who are not defined to RACF may use the system virtually without verification, unless, of course, they attempt to access data to which they are unauthorized.

Consider defining the following users to RACF:

- ▶ Interactive users of CICS, IMS, TSO/E, Tivoli NetView for z/OS, or other products that support logging on at a terminal
- ▶ Users who submit batch jobs
- ▶ z/OS or JES system operators
- ▶ Started procedures
- ▶ Node names in a network job entry (NJE)
- ▶ Remote job processing (RJP) or remote job entry (RJE) remote workstations or nodes
- ▶ Other users who can connect into z/OS using other subsystems like DB2, WebSphere Application Server, and NetView FTP.

### User identification

RACF uses an alphanumeric *user ID* for its user *identification*. The user ID identifies the person to the system as a RACF user. From a security point of view, the user ID is *unique* and must not be shared by different users. This uniqueness provides individual accountability.

In a client/server network environment, entities identify themselves using *digital certificates*. Therefore, we may have a one-to-one mapping between a client certificate and a RACF user ID. However, using RACF certificate name filtering allows many certificates to be assigned a single user ID with one profile.

## User verification

There are different techniques for user verification:

- ▶ Use a password phrase or password, something only the user *knows*

The system-encrypted password or password phrase is a character string that is known only by the user (not even by the security administrator) and, therefore, verifying the password or passphrase indicates to the system that the user is the actual person who owns that user ID. This can either be a password that is a maximum of eight characters long or a password phrase that is between nine (or 14) and 100 characters long. The password can use uppercase or mixed characters.

- ▶ Use something only the user *has*

This verification can be done with the use of a card with a magnetic stripe encoded with unique characters and used to verify the identity of a user to RACF on a z/OS system.

Normally, when you define a user to RACF, you assign a user ID *and* a temporary password. This then requires the user to change the password upon their first use. There are exceptions to this and RACF provides the **RESTRICTED** parameter, which we explain in 3.12, “RACF user attributes” on page 36.

Furthermore, you can have RACF installation exits that expand user verification.

Figure 3-12 shows the RACF user profile.

**Note:** It is the installation’s responsibility to accomplish and monitor security guidelines (for example, unique user IDs and password rules).

## 3.11 RACF user profile

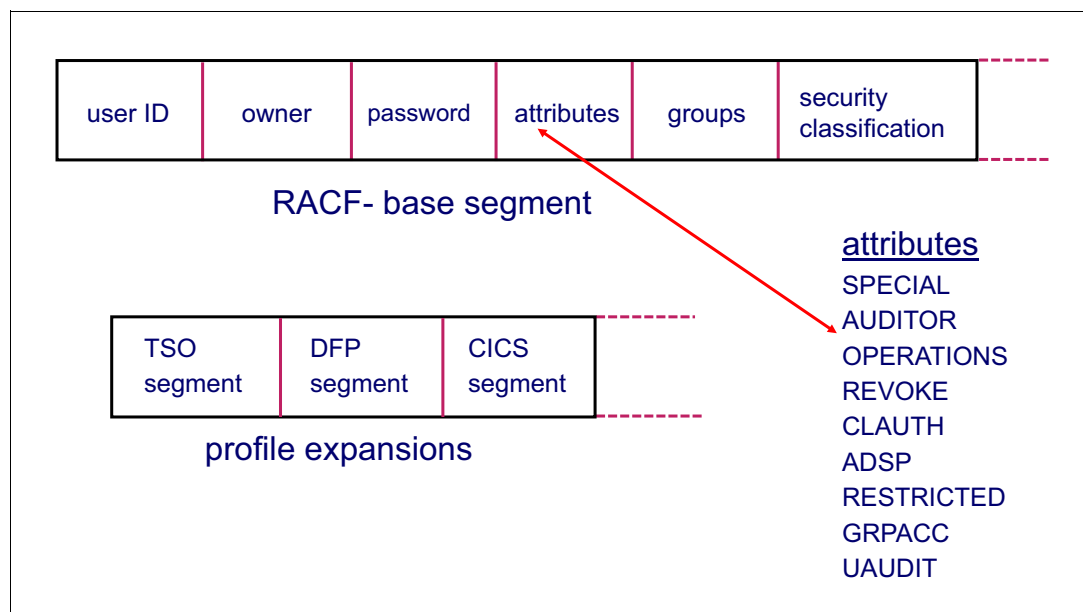


Figure 3-12 RACF user profile



## User profile

RACF stores information in its database. For each defined user ID, RACF keeps a *user profile* in the class USER. The profile consists of the RACF base segment and optionally additional segments that hold information related to the different resource managers.

Each segment of a user profile consists of fields. When you define a user's profile (using the **ADDUSER** command) or change a user's profile (using the **ALTUSER** command), you can specify the information contained in each field of each segment of the profile.

### RACF base segment

The base segment of a user profile contains basic information that is needed to define a user to RACF. Some of the more important fields are listed here:

<b>User ID</b>	The user ID is at the same time the name of the profile.
<b>Name</b>	The name of the user.
<b>Owner</b>	The <i>owner</i> of the profile has the authority to change the profile. Every profile in RACF needs an owner.
<b>Password</b>	The password entry is one-way <i>encrypted</i> . It is not possible to decrypt the password. If a user forgets the password phrase or password, the administrator has to set a new temporary password and the user must change this at the next logon.
<b>Attributes</b>	This field contains extraordinary attributes. The attributes SPECIAL, OPERATIONS, and AUDITOR should be given only to a few selected user IDs. Further information is provided in 3.12, "RACF user attributes" on page 36.
<b>DFLTGRP</b>	A user ID belongs at least to his <i>default group</i> , but can be a member of more groups. This field contains the default group that it is assigned to.
<b>CERTLABEL</b>	The certificate labels for the profiles in the DIGTCERT class that are associated with this RACF user ID.
<b>SECLEVEL</b>	User's installation-defined security level. It is where we can use the idea of levels to control access, that is, a user's level must be at least equal to or higher than the resource level it is trying to access.

Figure 3-13 on page 36 shows RACF user attributes.

**Important:** Ownership in RACF is of high importance. The owner of profiles can manipulate the profiles. For example, the owner can change or delete a profile. Your installation needs guidelines that define who is an owner of a profile.

## 3.12 RACF user attributes

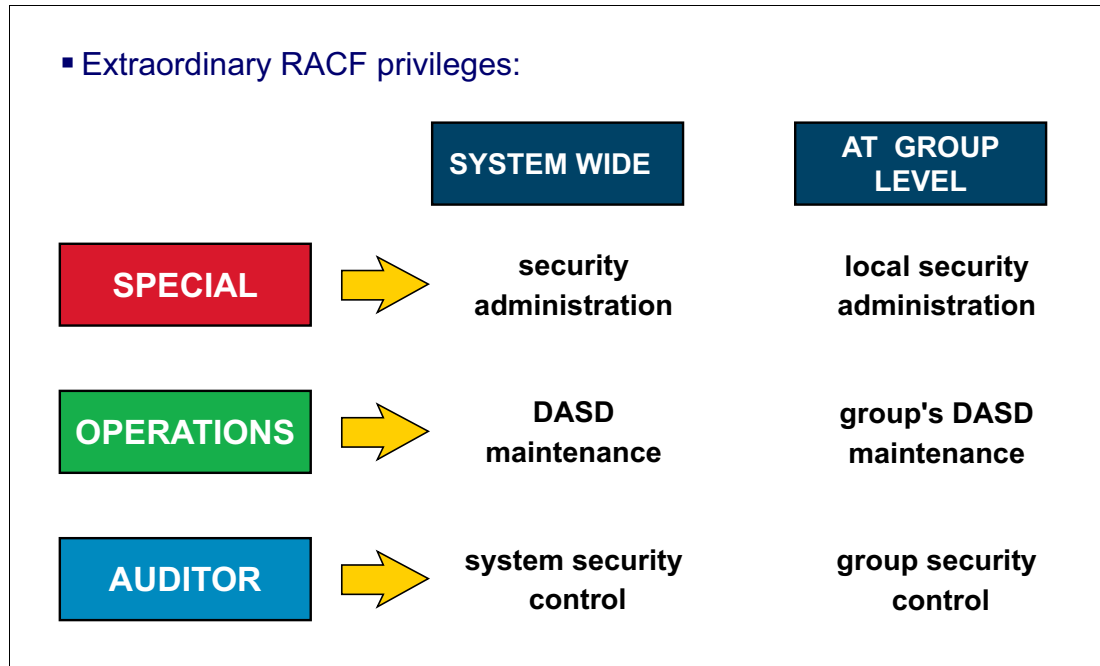


Figure 3-13 RACF user attributes

### User attributes

User attributes are extraordinary capabilities, limitations, or environments that can be assigned to a user either *system wide* or when the user is connected to a specific group or groups. When an attribute is to apply system wide, it is specified at the system level and is called a *user attribute*. When an attribute is to apply only to a specified group or groups, it is specified at the group level and is called a *group-related user attribute*.

User attributes that you specify in an **ADDUSER** or **ALTUSER** command are stored in the user's profile and are in effect regardless of the group to which the user is connected. However, attributes that you specify in a **CONNECT** command are valid only for this group.

The user attributes are as follows:

**SPECIAL** A user who has the SPECIAL attribute at the system level can issue all RACF commands and, therefore, is used only for special users, for example administrator. This attribute gives the user full control over all of the RACF profiles in the RACF database.

You can assign the SPECIAL attribute at the group level. When you do, the *group-SPECIAL* user has full control over all of the profiles within the *scope of the group*.

**Note:** Users with the SPECIAL attribute do not have access to all resources, but they can use commands to give themselves access to all resources. Because any user can access an unprotected resource, users who have the SPECIAL attribute should take care to protect their own data sets because they can contain sensitive information.

## AUDITOR

The AUDITOR attribute is given to users who are responsible for auditing RACF security controls and functions. To provide a check and balance on RACF security measures, you should give the AUDITOR attribute to security or group administrators other than those who have the SPECIAL attribute.

You can assign the AUDITOR attribute at the group level. When you do, the *group-AUDITOR* user's authority is limited to profiles that are within the scope of that group.

**Note:** Users with the AUDITOR attribute have the authority to specify logging options on the **ALTDSD**, **ALTUSER**, **RALTER**, and **SETROPTS** commands. In addition, the auditor can list auditing information using the **LISTDSD**, **RLIST**, **LISTUSER**, **LISTGRP**, and **SEARCH** commands, as well as the IRRUT100 utility.

## OPERATIONS

A user who has the system-wide OPERATIONS attribute has full access authorization to all RACF protected resources in the classes DATASET, DASDVOL, GDASDVOL, PSFMPL, TAPEVOL, VMBATCH, VMCMMD, VMMDISK, VMNODE, and VMRDR classes. (The last five classes in this list relate to RACF on z/VM.)

You can assign the OPERATIONS attribute at the group level. When you do, the *group-OPERATIONS* user's authority is limited to resources within the scope of that group.

**Note:** Because the OPERATIONS attribute can permit access to a wide range of resources, use this attribute very carefully. In some cases, you need to audit these users. So, caution must be exercised when assigning this attribute to a user as with the SPECIAL attribute.

## REVOKE

You can prevent a RACF user from entering the system by assigning the REVOKE attribute. This attribute is useful when you want to prevent a user from entering the system, but you can or will not use the DELUSER command because the user still owns RACF resource profiles.

You can also assign the REVOKE attribute on a group level by using the CONNECT command. If the user has the REVOKE attribute for a group, the user cannot enter the system by connecting to that particular group or access resources as a member of that group.

**Note:** RACF allows you to specify a future date for a REVOKE to occur (at both the system and the group level). You can also specify a future date to remove the REVOKE attribute by using the **RESUME** parameter on the **ALTUSER** command (for example, when you want to inhibit a user from entering the system during a long absence). If no date is specified on the **RESUME** parameter, the user is resumed immediately.

## CLAUTH

Users receive the CLAUTH attribute on a class-by-class basis. You cannot assign the CLAUTH attribute at the user or group level.

If a user has the CLAUTH attribute in a class, RACF allows the user to define profiles in that class.

**Note:** Give the CLAUTH attribute only to those users who are responsible for defining profiles to RACF in the specified classes and in any classes with the same POSIT value. This POSIT value is a way of assigning a ranking value of classes in the Dynamic CDT.

**RESTRICTED** You can prevent RACF users from gaining access to protected resources that they are not specifically authorized to access by assigning the RESTRICTED attribute on the **ADDUSER** or **ALTUSER** command.

**Note:** Restricted user IDs cannot be used to access protected resources that they are not specifically authorized to access. Access authorization for restricted user IDs bypasses global access checking. In addition, the UACC of a resource and an ID(\*) entry on the access list are not used to enable a restricted user ID to gain access.

**GRPCC** A user with this GROUP ACCESS attribute can define a group data set profile and as a consequence, other members of the group that this user belongs to have access to this profile.

**Note:** The group whose name is used as the high-level qualifier of the data set name is given UPDATE authority to the data set.

**ADSP** This is the automatic data set protection (ADSP) attribute. When you create a permanent data set or a tape data set, RACF automatically creates a discrete profile for it. It should be used sparingly because most data sets are protected by generic profiles.

Figure 3-14 shows RACF user segments.

### 3.13 RACF user segments

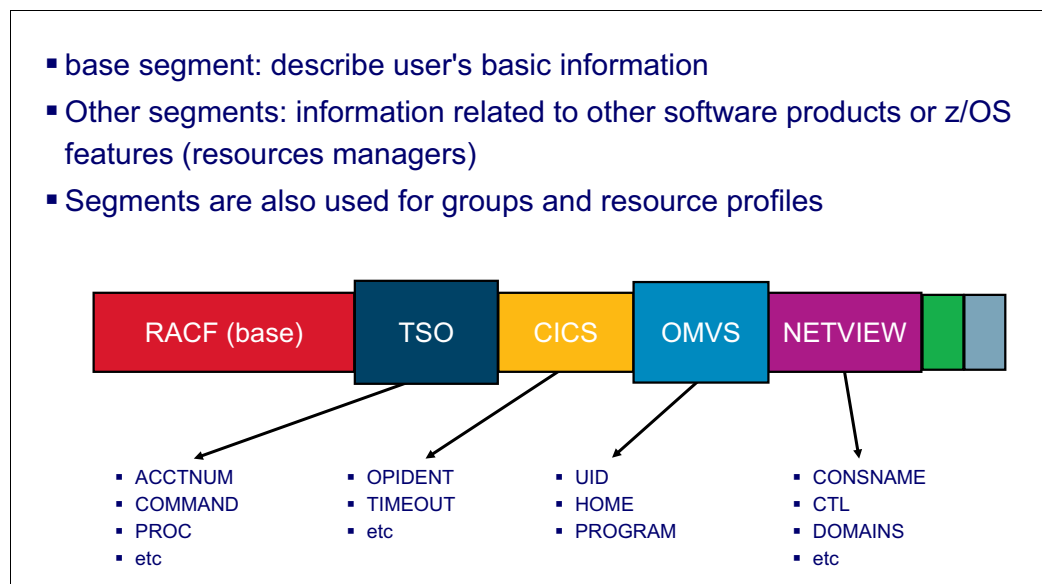


Figure 3-14 RACF user segments

When you define a user to RACF, you create a user profile in the RACF database. Each segment of a user profile consists of fields. When you define a user's profile (using the **ADDUSER** command) or change a user's profile (using the **ALTUSER** command), you can specify the information contained in each field of each segment of the profile. A user profile consists of a RACF base segment and optionally any of the following segments:

- ▶ CICS
- ▶ DCE
- ▶ CSDATA
- ▶ DFP
- ▶ KERB
- ▶ LANGUAGE
- ▶ LNOTES
- ▶ NDS
- ▶ NETVIEW
- ▶ OMVS
- ▶ OPERPARM
- ▶ OVM
- ▶ PROXY
- ▶ TSO
- ▶ WORKATTR

The base RACF segment is the part of the RACF profile that contains the fundamental information about a user, group, or resource and is common to several applications.

The other segments enable resource managers to keep related information.

The number of resource managers using RACF segments is continuously growing.

The following information is kept in the RACF base segment of the user profile:

<b>USERID</b>	User's identification
<b>NAME</b>	User's name
<b>OWNER</b>	Owner of the user's profile
<b>DFLTGRP</b>	User's default group
<b>AUTHORITY</b>	User's authority in the default group
<b>PASSWORD</b>	User's password (one-way encrypted)
<b>NOPASSWORD</b>	Gives the user the PROTECTED attribute when the user has the NOPHRASE and NOOIDCARD attributes
<b>PHRASE</b>	Optionally a Password Phrase (one-way encrypted)
<b>NOPHRASE</b>	Indicates that the user cannot enter the system using a password phrase and when the user also has the NOPASSWORD and NOOIDCARD attributes, gives the user the PROTECTED attribute
<b>REVOKE</b>	Date on which RACF prevents the user from having access to the system
<b>RESUME</b>	Date on which RACF lets the user have access to the system again
<b>UACC</b>	Default universal access authority for resources that the user defines
<b>WHEN</b>	Days of the week and hours of the day during which the user has access to the system
<b>ADDCATEGORY</b>	User's installation-defined security category
<b>SECLEVEL</b>	User's installation-defined security level

<b>CLAUTH</b>	Classes in which the user can define profiles
<b>SPECIAL</b>	Gives the user the system-wide SPECIAL attribute
<b>AUDITOR</b>	Gives the user the system-wide AUDITOR attribute
<b>OPERATIONS</b>	Gives the user the system-wide OPERATIONS attribute
<b>DATA</b>	Installation-defined data
<b>ADSP</b>	Indicates that all permanent data sets the user creates are to be RACF protected with discrete profiles
<b>GRPACC</b>	Indicates that other group members can have access to any group data set the user protects with a data set profile
<b>MODEL</b>	Name of the data set model profile to be used when creating new data set profiles, either generic or discrete
<b>OIDCARD</b>	Indicates that the user must supply an operation ID card when logging on to the system
<b>RESTRICTED</b>	Indicates that global access checking, the ID(*) entry on the access list, and the UACC will not be used to allow this user access to a protected resource.
<b>SECLABEL</b>	User's default security label
<b>CERTNAME</b>	The names of the profiles in the DIGTCERT class that are related this RACF user ID
<b>CERTLABL</b>	The certificate labels associated with the profiles in the DIGTCERT class that are related to this RACF user ID
<b>CERTPUBK</b>	The public key associated with a public key certificate. This is the BER-encoded public key as specified in the certificate. BER means basic encoding rules to cover encoding specific data to an octet stream for transmission.
<b>CERTSJDN</b>	The subject name of the entity to whom the certificate is issued. This is the BER-encoded format of the subject's distinguished name as contained in the certificate. A distinguished name is a set of fields to uniquely identify the certificate owner.
<b>NMAPNAME</b>	The names of the profiles in the DIGTNMAP class containing certificate name filters that are associated with this RACF user ID, for example, you can filter by some or all of the fields in the distinguished name.
<b>NMAPLABL</b>	The labels for the certificate name filters that are associated with this RACF user ID.

Figure 3-15 on page 41 shows RACF user ID and password information.

## 3.14 RACF user ID and password

- Password management
  - Allows user to select own password phrase and/or password
  - Only user knows his password phrase and/or password
  - Security administrator cannot read, but can reset password and password phrase
- Password and password phrase control
  - Interval, history, syntax rules, expiration
  - warning, suppression
  - Last logon message
  - Revoke invalid attempts
  - DES one-way encryption
  - EXIT - check or generate passwords
- Alternates to password verification

Figure 3-15 RACF user ID and password

User identification is achieved using the *user ID*, which is a string of characters that uniquely identifies a user to a system.

In RACF, users select their own password (and optionally a password phrase) and only the user knows these values. If a password or password phrase needs to be reset, the security administrator either resets it to the default or sets a temporary password (and optionally a password phrase). This profile is normally in an expired state, thus forcing the user to enter a new password or password phrase on the first logon.

You can set a variety of rules for forming valid passwords, using the **SETROPTS** command (for system-wide settings) or the **PASSWORD** command (to affect only one user). You can change such things as the number of days a password is valid, how long to maintain password history to prevent the user from reusing the same password again, and so on. There is a detailed explanation of setting up a rule in, “**SETROPTS PASSWORD(RULE1(LENGTH(8) VOWEL(1,3,5:8) NUMERIC(2,4)))**” on page 79.

The password and password phrase is one-way encrypted using a Data Encryption Standard (DES) algorithm. The key being used is the password itself. The encrypted password and password phrase are stored in the user profile. Remember that RACF never looks at the raw password, for checking it compares the encrypted forms. This way the only individual that knows the raw password is the user.

To give the user feedback each time that they log on to TSO/E, they receive information when they last logged on. The user checks to see if this is correct because it might indicate an invalid use of their user ID. They will see this message:

```
ICH70001I USER001 LAST ACCESS AT 11:28:24 ON TUESDAY, JULY 23, 2013
```

## Alternatives to password verification

The following alternatives are available for password verification:

1. RACF using its Secured Signon function allows workstations and client machines in a client/server environment to use a *PassTicket* in place of a password. A PassTicket can be generated by an authorized routine in z/OS or on any other platform. The creator of the PassTicket and the verifier of the PassTicket must share a “common secret.” In addition, the creator and verifier must have the same user ID, application name, and time. The PassTicket is valid for about 10 minutes. You can enforce that a PassTicket is only valid for one logon. It removes the need to send RACF passwords and password phrases across the network in clear text.
2. RACF allows the use of an operator identification card (OIDCARD) in place of, or in addition to, the password during terminal processing. By requiring that a person not only know a password but also furnish an OIDCARD, an installation has increased assurance that the user ID was entered by the proper user.
3. z/OS UNIX users are also identified with numeric UNIX user identifiers (UIDs), and z/OS UNIX groups are identified with numeric UNIX group identifiers (GIDs). Unlike user names or group names, these numeric IDs can be shared by more than one user. However, this practice is not recommended.
4. When you combine a driving application, such as z/OS HTTP Server with middleware that supports a secure protocol, such as SSL, and the secure certificate management functions of RACF, you can implement a secure certificate environment on z/OS. It will pass the client’s digital certificate to RACF. Thus, the RACF user ID and password of each client do not need to be supplied when accessing web pages. The certificate must be mapped to a RACF user ID in order for authorization checking when operating with the z/OS operating system. It can be a 1:1 or 1:many mapping, the latter case covering many individual clients performing possibly the same function.

Figure 3-16 on page 43 shows the process of adding a new user to the RACF database.



## 3.15 Adding a new user to RACF

- Add a new user:
  - **ADDUSER JAMES NAME('BROWN JAMES') DFLTGRP(MFG) OWNER(ADMUSERS) PASSWORD(NEW2DAY)**
- List the user:
  - **LISTUSER JAMES**

**O  
U  
T  
P  
U  
T**

```
USER=JAMES  NAME=BROWN JAMES          OWNER=ADMUSERS  CREATED=13.205
DEFAULT-GROUP=MFG      PASSDATE=00.000  PASS-INTERVAL= 90  PHRASEDATE=N/A
ATTRIBUTES=NONE
REVOKE DATE=NONE      RESUME DATE=NONE
LAST-ACCESS=UNKNOWN
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED  (DAYS)          (TIME)
-----
ANYDAY                                ANYTIME
GROUP=MFG      AUTH=USE        CONNECT-OWNER=ADMUSERS  CONNECT-DATE=13.205
CONNECTS=      00  UACC=NONE    LAST-CONNECT=UNKNOWN
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE  RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
NONE SPECIFIED
SECURITY-LABEL=NONE SPECIFIED
```

Figure 3-16 Adding a new user to the RACF database

### How to add a user

When you define a user's profile (using the **ADDUSER** command) or change a user's profile (using the **ALTUSER** command), you can specify the information contained in each field of each segment of the profile.

The command adds a profile for the new user to the RACF database and creates a connect profile that connects the user to whichever default group you specify. The user profile consists of a RACF base segment and, optionally, other segments such as a TSO segment, a DFP segment, or an OMVS segment. You can use this command to define information in any segment of the user's profile.

Figure 3-16 shows sample output from the following **ADDUSER** command when the **LISTUSER** is issued:

```
ADDUSER JAMES NAME('BROWN JAMES') DFLTGRP(MFG)
OWNER(ADMUSERS) PASSWORD(NEW2DAY)
```

This command adds a new user ID, **JAMES**, into default group, **MFG**.

Figure 3-17 on page 44 shows the process of resetting a user password.

## 3.16 Reset a user password

- How to reset a Password :

- List the user : LISTUSER JAMES

- ALU James RESUME PASS(NEW PASSWORD) => If REVOKED
- ALU James PASS(new password) => If not REVOKED
- ALU James PASS(new password)NOEXPIRED => If not REVOKED

```

O
U
T
P
U
T
USER=JAMES  NAME=BROWN JAMES          OWNER=ADMUSERS  CREATED=13.205
DEFAULT-GROUP=MFG      PASSDATE=00.000  PASS-INTERVAL= 90  PHRASEDATE=N/A
ATTRIBUTES=REVOKED      ← this user has been revoked
REVOKE DATE=NONE      RESUME DATE=NONE
LAST-ACCESS=UNKNOWN
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED      (DAYS)          (TIME)
-----
ANYDAY
GROUP=MFG          AUTH=USE          ANYTIME
CONNECT-OWNER=ADMUSERS  CONNECT-DATE=13.205
CONNECTS=          00  UACC=NONE      LAST-CONNECT=UNKNOWN
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE      RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
NONE SPECIFIED
SECURITY-LABEL=NONE SPECIFIED

```

Figure 3-17 Resetting a password

A system administrator is often asked to reset a user's password. There are two common reasons for resetting a password:

1. The user forgot the password (or made too many errors when attempting to change it).
2. The user ID was REVOKED for some other reason, for example, revoked while on vacation.

You can use the RACF ISPF panels to reset passwords but it is easier to use the following commands:

**PASSWORD** When used to reset another user's password, the only option is to set the password equal to the user's default group name. The default group name is often SYS1. So, if the **PASSWORD** command is used to reset a user's password, the password is probably *SYS1*, which has obvious security consequences.

**ALTUSER** You set the password phrase or the password. You can also specify whether the user must specify the passwords again. This is indicated by EXPIRED or NOEXPIRED.

In both cases, the password is marked automatically as *expired*, by default. Thus, the user is forced to select a new password when logging on to the system the next time. With the **ALU** command, you can also set an unexpired password, which is the password that the user can use until changing it for some other reason.

Before resetting a password, we suggest that you always use the **LISTUSER** command to verify that the user definition exists and to determine if the user is **REVOKED**. For example, we can use this command:

```

ALU martin RESUME PASS(newpwd)      <== if REVOKED
ALU martin PASS(newpwd)             <== if not REVOKED
ALU martin PASS(newpwd) NOEXPIRED    <== if not REVOKED
PASSWORD NOINTERVAL USER(MARTIN)    <== if you want this to set password

```

You need to tell Martin the new password that you assigned. Martin needs the new password to log on but is forced to change the password immediately to a password of his own selection (unless you used the **NOEXPIRED** option). The **PASSWORD NOINTERVAL** command prevents this user's password from ever expiring and sets the password to the user ID's default group name, which is a possible security exposure and should be avoided. You need **SPECIAL** authority to issue these commands.

### How to reset a password with ISPF panels

You can also use the RACF ISPF panels to change or reset passwords. The end result is the same as using the direct commands discussed previously.

The path to the appropriate RACF ISPF panels is:

```

ISPF Primary Option Menu
  RACF      (select RACF from the primary ISPF menu)
    RACF - Services Option Menu
      User Profiles and Your Own Password
        RACF - User Profile Services( enter target userid in the User field
          CHANGE (enter S in Change PASSWORD related information field

```

When the panel that is shown in Figure 3-18 displays, carry on from this point.

```

RACF - CHANGE USER JAMES PASSWORD RELATED FIELDS
COMMAND ==>

- use a default PASSWORD
  or change PASSWORD (case sensitive)
  ==>          <=== User's password
  ==>          <=== Re-enter password to verify

- NOPHRASE to clear the current PHRASE
  or change PHRASE (case sensitive)
  ==>
                                     <=== Up to 100 characters in quotes
  ==>
                                     <=== Re-enter phrase to verify

EXPIRED  ___   YES to mark new password and phrase as expired?

INTERVAL ___   1 - 254 days, NO, or blank

REVOKE   _____ YES, NO, mm/dd/yy (date) or blank
RESUME   _____ YES, NO, mm/dd/yy (date) or blank

```

Figure 3-18 RACF Change user JAMES

Remember that the password that you assign must be changed by the user when that user logs on to the system the next time. You can use this same panel, and other panels that display after you press Enter, to change the same elements as the **ALTUSER** command.

Figure 3-19 shows how to alter a user ID.

### 3.17 Alter a user ID

- Alter a user: **ALTUSER JAMES AUDITOR**
- List the user: **LISTUSER JAMES**

**USER=JAMES NAME=BROWN JAMES OWNER=ADMUSERS CREATED=13.205**  
**DEFAULT-GROUP=MFG PASSDATE=00.000 PASS-INTERVAL= 90 PHRASEDATE=N/A**  
**ATTRIBUTES=AUDITOR**  
**REVOKE DATE=NONE RESUME DATE=NONE**  
**LAST-ACCESS=13.209/12:39:18**

**User attribute changed  
after ALTUSER command**

**O  
U  
T  
P  
U  
T**

Figure 3-19 Altering a user ID

Use the **ALTUSER** command to change the information in a user's profile, including the user's system-wide attributes and authorities. The user profile consists of a RACF base segment and, optionally, other segments such as a TSO segment or a DFP segment. You can use this command to change information in any segment of the user's profile.

When you change a user's level of authority in a group (using the **AUTHORITY** parameter), RACF updates the appropriate group profile. When you change a user's default universal access authority for a group (using the **UACC** operand), RACF changes the appropriate connect profile. For all other changes, RACF changes the user's profile.

**Note:** If the user is currently logged on, changes to the attributes (except for **OWNER** and **AUTHORITY**) do not take effect until the next time the user logs on, even though the **LISTUSER** command shows the new values.

Figure 3-19 shows sample output from the following **ALTUSER** command, which adds the attribute of **AUDITOR** to the user ID **JAMES**:

```
ALTUSER JAMES AUDITOR
```

Figure 3-20 on page 47 shows the process of changing a user password interval.

## 3.18 Change a user password interval

- Change password interval:
  - PASSWORD USER(JAMES) INTERVAL(60)
- List the user: LISTUSER JAMES

O U T P U T	USER=JAMES NAME=BROWN JAMES OWNER=ADMUSERS CREATED=13.205
	DEFAULT-GROUP=MFG PASSDATE=00.000 PASS-INTERVAL= 60 PHRASEDATE=N/A
	ATTRIBUTES=NONE
	REVOKE DATE=NONE RESUME DATE=NONE
	LAST-ACCESS=13.209/12:39:18
	CLASS AUTHORIZATIONS=NONE

ALTUSER change shows Interval  
now at 60 days

Figure 3-20 Change a user password interval

The interval indicates the number of days during which a password remains valid. The range is from one through 254 days.

The value that you specify here cannot exceed the value, if any, that your installation has specified using the INTERVAL operand on the **SETROPTS** command. The initial system default after RACF initialization is 30 days.

If you specify INTERVAL on the **PASSWORD** command without a change-interval value, RACF uses the system interval value (if any) that your installation specified or the system default.

Figure 3-20 shows sample output from the following **PASSWORD** command, which sets the password expiration date for user ID James to 60 days:

```
PASSWORD USER(JAMES) INTERVAL(60)
```

To override the current system default password expiry interval, you would use the **SETROPTS** command and specify a new value using the **INTERVAL** parameter.

Figure 3-21 on page 48 shows the process of deleting a user from the RACF database.

## 3.19 Delete a user ID

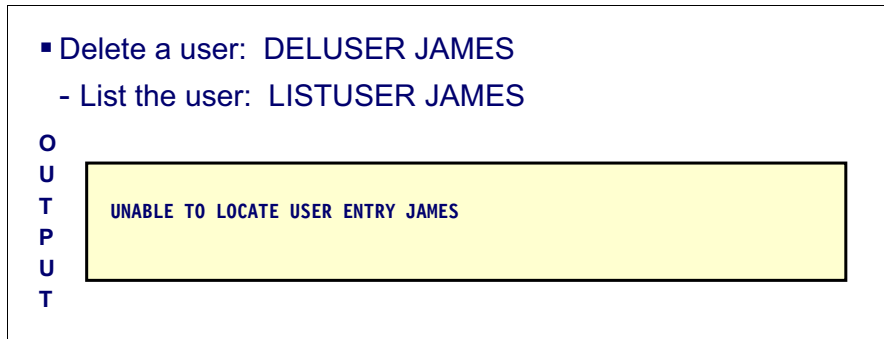


Figure 3-21 Deleting a user from the RACF database

Use the **DELUSER** command to delete a user from RACF. This command removes the user's profile and all user-to-group connections for the user. (The connect profiles define the user's connections to various RACF groups.)

There are, however, other places in the RACF database where the user's user ID might still appear. The **DELUSER** command does not delete the user ID from all these places. Specifically, the user could be the owner of a group, the owner of a user's profile, the owner of a group data set, or in an access list for any resource. Before issuing **DELUSER**, you must first issue the **REMOVE** command to assign new owners for any group data sets the user owns in groups other than his default group. You can use the RACF Remove ID utility (IRRRID00) to remove all of the occurrences of a user ID. For information about using the RACF Remove ID utility, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

To use the **DELUSER** command, at least one of the following scenarios must be true:

- ▶ You must have the **SPECIAL** attribute.
- ▶ The user profile to be deleted must be within the scope of a group in which you have the group-**SPECIAL** attribute.
- ▶ You must be the owner of the user's profile.

Figure 3-21 shows sample output from the following **DELUSER** command, which deletes the user ID James.

```
DELUSER JAMES
```

Figure 3-22 on page 49 shows the user-related RACF commands.

## 3.20 User-related RACF commands

- ADDUSER
- ALTUSER
- CONNECT
- DELUSER
- REMOVE
- LISTUSER
- PERMIT
- PASSWORD
- RACMAP

Figure 3-22 User-related RACF commands

You define users to RACF by issuing RACF commands that include various user attributes, as well as other control information that RACF uses. You might use some of the following commands in your user-definition tasks:

<b>ADDUSER</b>	Add a user profile to RACF.
<b>ALTUSER</b>	Change a user's RACF profile.
<b>CONNECT</b>	Connect a user to a group.
<b>DELUSER</b>	Delete a user profile from RACF and remove connection to a group.
<b>REMOVE</b>	Remove a user from a group and assign a new owner for group data sets owned by the removed user.
<b>LISTUSER</b>	Display the contents of a user's profile.
<b>PERMIT</b>	Permit a user to access a resource (or deny access to a resource).
<b>PASSWORD</b>	Change a user's password.
<b>RACMAP</b>	Create and maintain a mapping association between a RACF user ID and one or more distributed user identities.

In addition to defining individual users, you can define groups of users. Group members can share common access authorities to a protected resource.

Figure 3-23 on page 50 explains RACF groups and advantages.

## 3.21 RACF groups

- Group = collection of users
  - Every user belongs to 1 (or more) group
  - Groups can correspond to department, organization, function, product, etc.
  - Resultant "tree" structure of related groups
- Group advantages:
  - Reduces administrative efforts
  - Allows decentralized administration by delegation of administrative authority

Figure 3-23 RACF groups

### RACF groups

With RACF, all defined users belong to at least one group. You can think of the groups forming a hierarchical, or "tree" structure, where each group is owned by a superior group. Groups can also own resources as well as users in another group.

RACF has the following types of groups:

**Administrative** You can create a group simply as an administrative convenience. For example, you might create a group to represent an organizational entity, such as a region or a division. With RACF delegation, you can create this kind of group for each group administrator. Operating from such groups, the group administrators can then define other groups needed by their local users.

**Holding** This is a technique that retains user definition centrally, yet allows the effective use of group administrators to establish a holding group. You define all users centrally and initially connect them to a group named HOLD with the minimum of authorities. HOLD does not appear in any access lists and, therefore, has no real significance to the user.

Group administrators, to whom you give CONNECT (but not JOIN) authority, can connect the appropriate users to the groups under their control and change the user's default group name as appropriate. This technique allows the installation to assign correct account numbers (in TSO segment) and control other installation considerations while allowing flexibility in the grouping of the user population:

**Data Control** You can create a group to act as a control point for the protection of data. For example, by using the group SYS1, you can determine which users are permitted to protect the SYS1 data sets. Only users with CREATE authority or higher in this group can protect system data sets. At your location, you might consider defining one such group for every high level qualifier representing data that is to be protected.

**Functional** A group can represent a functional area of the installation for the purpose of data sharing. For example, a financial analyst might need to access various resources across many groups, such as accounting, payroll, marketing, and others. Of course, the owners of each resource



could permit the financial analyst to access their resources by placing the analyst's user ID on an access list. But if a new financial analyst takes over the job, it is then necessary to add the new user ID to each RACF profile. Likewise, the RACF profiles must be updated when the analyst no longer has a need to access the data. This arrangement involves a great deal of unnecessary activity by the resource owners.

Instead, you can create a group that represents the financial analyst function and permits access to the data defined to the group. Access to the entire range of data can then be managed by controlling the user population in the defined group. For cases involving one-time access, owners of the needed data would simply PERMIT access by the defined group. Where appropriate, the group name could be included in profile access lists to ensure automatic availability of needed data to the financial analyst group. New financial analysts could be connected to the group as required to gain access to the entire range of data. Likewise, analysts could be removed from the group whenever necessary. By controlling the user population of such a functional group, resource profile changes on a day-to-day basis becomes unnecessary.

## User

You can define a group to serve as an anchor point for users who otherwise have no common access requirements. For example, engineers and scientists, as well as other problem-solving users, might have no need to access application-related data in the system. Their only interest might be in their own personal data. You can place this set of users in a single group that has no access to other data.

You can also define groups based on access level. For example, if PAY.DATA is a RACF defined data set, two groups could be defined, PAYREAD and PAYUPDTE, both of which would appear in the PAY.DATA access list, but with READ and UPDATE access, respectively. Any users requiring access would be connected as appropriate, by the group administrator.

Figure 3-24 on page 52 shows a RACF group structure example.

## 3.22 RACF group structure example

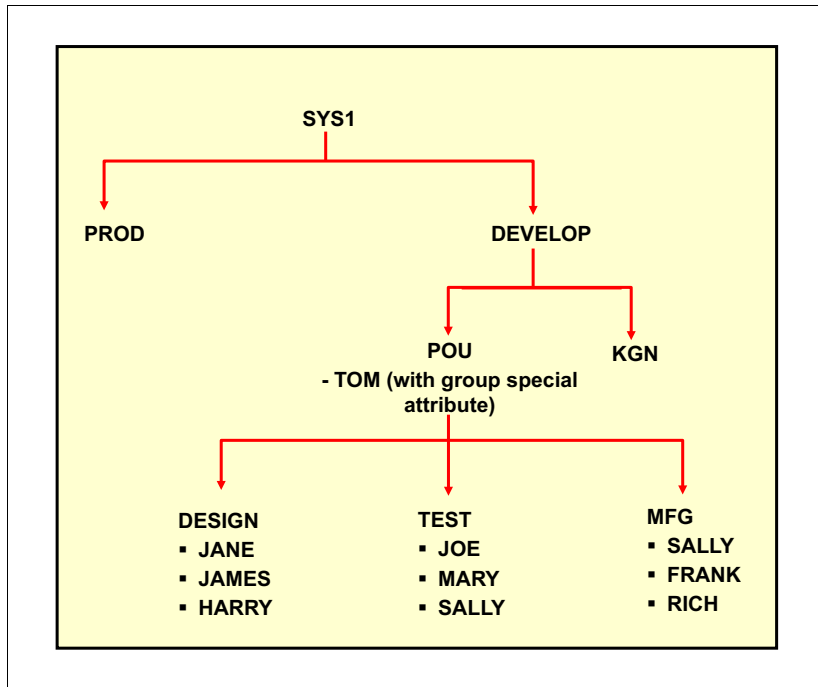


Figure 3-24 RACF group structure example

The group structure of RACF can be mapped to the organizational structure that exists at your installation. That is, RACF conforms naturally to a tree structure of groups, where each group (except SYS1, which is predefined as the highest group) has a superior, or owning, group. Groups can correspond directly to business entities such as divisions, departments, and projects. Users can be connected to one or more groups.

When you define a group, consider the basic purpose of the group. Is it an administrative group, a holding group, a data control group, a functional group, or a user group? When setting up RACF groups, keep in mind that the maximum number of users that you can connect to any one group is approximately 5900.

You should map your groups to your organization's structure and arrange them hierarchically, with the IBM supplied SYS1 group as the highest group so that each group is a subgroup of another group.

A user can be connected in more than one group (in Figure 3-24, SALLY is connected to MFG and TEST groups).

In Figure 3-24, DESIGN, TEST, and MFG are all owned by group POU. Tom is connected to group POU as SPECIAL, which gives Tom (who is the RACF administrator) control over all POU resources DESIGN, TEST, and MFG.

Figure 3-25 on page 53 shows the process of adding a group.

## 3.23 RACF group-related commands: Add a group

- Add a group:
  - ADDGROUP EXPED OWNER(ADMGRPS) SUPGROUP(POU)
- List the group:
  - LISTGRP EXPED

O  
U  
T  
P  
U  
T

```
INFORMATION FOR GROUP EXPED
SUPERIOR GROUP=POU          OWNER=ADMGRPS   CREATED=13.209
NO INSTALLATION DATA
NO MODEL DATA SET
TERMUACC
NO SUBGROUPS
NO USERS
```

Figure 3-25 Add a group

Use the **ADDGROUP** command to define a new group to RACF. The command adds a profile for the new group to the RACF database. It also establishes the relationship of the new group to the superior group that you specify.

Group profiles consist of a RACF base segment and, optionally, other segments such as DFP and OMVS. You can use this command to specify information in any segment of the profile.

To use the **ADDGROUP** command, you must meet at least one of the following conditions:

- ▶ Have the SPECIAL attribute
- ▶ Have the group-SPECIAL attribute and the superior group is within your group-SPECIAL scope
- ▶ Be the owner of the superior group
- ▶ Have JOIN authority in the superior group

Figure 3-25 shows sample output from the **ADDGROUP** command, adds a new group named EXPED, and is a subgroup to group POU:

```
ADDGROUP EXPED OWNER(ADMGRPS) SUPGROUP(POU)
```

Figure 3-26 on page 54 shows sample output from the **ALTGROUP** command.

## 3.24 RACF group-related commands: Alter a group

```

O
U
T
P
U
T

  ■ Alter a group:
    - ALTGROUP EXPED SUPGROUP(KGN)
  ■ List the group:
    - LISTGRP EXPED

  INFORMATION FOR GROUP EXPED
  SUPERIOR GROUP=KGN      OWNER=ADMGRPS      CREATED=13.209
  NO INSTALLATION DATA
  NO MODEL DATA SET
  TERMUACC
  NO SUBGROUPS
  NO USERS

  KGN is now the new superior Group

```

Figure 3-26 Alter a group

Use the **ALTGROUP** command to change various fields within the segments in the selected profile:

- ▶ The superior group of a group
- ▶ The owner of a group
- ▶ The terminal indicator for a group
- ▶ A model profile name for a group
- ▶ The installation-defined data associated with a group
- ▶ The default segment information for a group (for example, DFP or OMVS)

To change the superior group of a group, you must meet at least one of the following conditions:

- ▶ You must have the **SPECIAL** attribute
- ▶ All the following group profiles must be within the scope of a group in which you have the group-**SPECIAL** attribute:
  - The group whose superior group you are changing
  - The current superior group
  - The new superior group
- ▶ You must be the owner of, or have **JOIN** authority in, both the current and the new superior groups.

**Note:** You can have **JOIN** authority in one group and be the owner of, or have the group-**SPECIAL** attribute in, the other group.

Figure 3-26 shows sample output from the **ALTGROUP** command, which moves the group named **EXPED** from being a subgroup of group **POU** to a subgroup to group **KGN**:

```
ALTGROUP EXPED SUPGROUP(KGN)
```

Figure 3-27 on page 55 shows sample output from the **DELGROUP** command.

## 3.25 RACF group-related commands: Delete a group

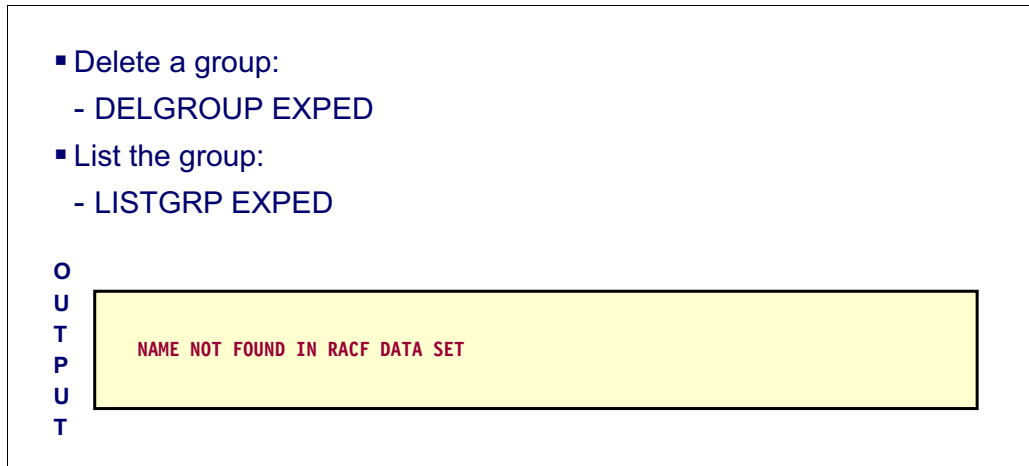


Figure 3-27 Delete a group

Use the **DELGROUP** command to delete a group and its relationship to its superior group from RACF.

There are, however, other places in the RACF database where the group name might appear, and DELGROUP processing does not delete these other occurrences of the group name. For example, the group name could be in the access list for any resource. You can use the RACF Remove ID utility (IRRRID00) to remove all occurrences of a group name. For information about using the RACF Remove ID utility, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

Figure 3-27 shows sample output from the **DELGROUP** command, which deletes the EXPED group:

```
DELGROUP EXPED
```

Figure 3-28 on page 56 shows sample output from the **CONNECT** command.

## 3.26 Connect a user to a group

- Connect the user to a group:
  - CONNECT JAMES GROUP(TEST)
- List the user: LISTUSER JAMES

```
O
U
T
P
U
T
USER=JAMES  NAME=BROWN JAMES          OWNER=ADMUSERS  CREATED=13.205
DEFAULT-GROUP=MFG    PASSDATE=00.000  PASS-INTERVAL= 60  PHRASEDATE=N/A
ATTRIBUTES=NONE
...
...
GROUP=MFG      AUTH=USE  CONNECT-OWNER=ADMUSERS  CONNECT-DATE=13.205
CONNECTS=     00  UACC=NONE    LAST-CONNECT=UNKNOWN
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE  RESUME DATE=NONE
GROUP=TEST     AUTH=USE  CONNECT-OWNER=ADMUSERS  CONNECT-DATE=13.209
CONNECTS=     00  UACC=NONE    LAST-CONNECT=UNKNOWN
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE  RESUME DATE=NONE
...
```

Figure 3-28 Connecting a user to a group

Use the **CONNECT** command to connect a user to a group, modify a user's connection to a group, or assign the group-related user attributes. If you are creating a connection, defaults are available as stated for each operand. If you are modifying an existing connection, no defaults apply.

To use the **CONNECT** command, one of the following conditions must be true:

- ▶ The **SPECIAL** attribute
- ▶ The group-**SPECIAL** attribute in the group
- ▶ The ownership of the group
- ▶ **JOIN** or **CONNECT** authority in the group

Figure 3-28 shows sample output from the **CONNECT** command, which connects user James to group TEST:

```
CONNECT JAMES GROUP(TEST)
```

Figure 3-29 on page 57 shows sample output from the following **REMOVE** command.

## 3.27 Remove a user from a group

- Remove a user from a group: REMOVE JAMES GROUP(TEST)
- List the user: LISTUSER JAMES

```
O
U
T
P
U
T
USER=JAMES NAME=BROWN JAMES OWNER=ADMUSERS CREATED=13.205
DEFAULT-GROUP=MFG PASSDATE=00.000 PASS-INTERVAL= 60 PHRASEDATE=N/A
ATTRIBUTES=NONE
REVOKE DATE=NONE RESUME DATE=NONE
LAST-ACCESS=13.209/12:39:18
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED (DAYS) (TIME)
-----
ANYDAY ANYTIME
GROUP=MFG AUTH=USE CONNECT-OWNER=ADMUSERS CONNECT-DATE=13.205
CONNECTS= 00 UACC=NONE LAST-CONNECT=UNKNOWN
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
NONE SPECIFIED
```

Figure 3-29 Removing a user from a group

You can use the **REMOVE** command to remove a user from a group, and to assign a new owner to any group data set profiles that the user owns on behalf of that group.

To use the **REMOVE** command, one of the following conditions must be true:

- ▶ The SPECIAL attribute
- ▶ The group-SPECIAL attribute in the group
- ▶ The ownership of the group
- ▶ JOIN or CONNECT authority in the group

Figure 3-29 shows sample output from the following **REMOVE** command:

```
REMOVE JAMES GROUP(TEST)
```

Figure 3-30 on page 58 shows data sets and general resources.

## 3.28 Data sets and general resources

- Classes of resources profiles:
  - Data set
    - Tape data set
    - DASD data set
  - General resources
    - Terminals
    - Programs
    - IMS transactions
    - etc
- Three types of profiles:
  - DISCRETE profiles
  - GENERIC profiles
  - GROUPED profiles

Figure 3-30 Data sets and general resources

### Resource profile types

To protect a general resource, create a general resource profile using the **RDEFINE** command. When you create a general resource profile, you must specify a general resource class for the profile. IBM supplies a list of the general resource classes in the dynamic class descriptor table (CDT). The classes for z/OS systems are relevant to the system on which you are running the z/OS Security Server (RACF).

RACF protected resources can be divided into two categories: Data sets and general resources. General resources are all of the resources that are defined in the class descriptor table. For example, general resources include DASD and tape volumes, load modules (programs), terminals, and others.

RACF allows the installation to set its own rules for controlling the access to its resources by defining what is controlled at what level. The installation can tailor RACF to interact with its present operating environment and assign security responsibilities either on a system-wide or a group-wide basis.

Your installation can add new CDT entries or modify or delete existing entries that you have added in the installation-defined class descriptor table (ICHRRCDE). When you define a new resource class, you can optionally designate that class as either a resource group class or a resource member class. For a resource group class, each user or group of users that is permitted access to that resource group is permitted access to all members of the resource group. For each resource group class that you create, you must also create a second class that represents the members of the group.



It is possible to define dynamic CDT entries. This is done by defining profiles in the CDT class using the **RDEFINE** and **RALTER** commands.

### Types of profiles

We have seen what components and some of the fields we can find in a profile. We now look at the profile and the general resource that it is protecting:

<b>Generic</b>	This a profile where we use a masking character to allow one profile to protect many resources. For example, if we had many data sets on the system, which were called <b>FEDERAL.ACCOUNTS.*</b> we use that name to make a generic profile to protect all those data sets.
<b>Discrete</b>	This is where we want to protect a specific resource. This would be used in special circumstances because many discrete profiles would add to the administrative overhead.
<b>Group</b>	A resource group profile where the resources to be protected are added to this group profile via the <b>ADDMEM</b> parameter. This group profile name does not need to match the resources that it protects but it has a list of resources it does protect.

Figure 3-31 describes resource profiles.

## 3.29 More on profiles for data sets and general resources

- Profiles contain:
  - The owner of the profile
  - The auditing parameters
  - The Universal Access authority
  - An access list with users and groups
  - A "warning" indicator
  - A security classification
  - A real-time notification information
  - An erase-on-scratch indication for data sets
  - A volume & a unit (if data set)
  - A security retention period (if tape data set)

Figure 3-31 Resource profiles

Resource profiles are different from user profiles because resource profiles need to hold characteristic information about the resources they protect. Here is a selected number of fields from this type of profile to show its characteristics:

**The profile owner** This could be a user or a group name. This group could well be the profile associated with a specific application. This enables us to arrange data set protection on a functional basis.

**Auditing parameters** Here we specify what access attempts and also access level attempts that we want records. Access levels come into use when we use access levels to govern access.

**Universal Access Authority**

This is commonly called *UACC* and is best set to NONE. This makes access by using this as not possible unless specific access is given in access lists at group or user level.

**An access list with users and groups**

This is indirectly obtained from the owner and if this new resource is built modeled of an existing resource.

**A “Warning” indicator**

An interim measure where RACF permits access even though access authority is insufficient. This setting would draw the attention of any auditor. A warning message is issued when access occurs.

**Security classification**

This covers a range of fields from security category, security label, and security level. These are installation defined.

**A real-time notification information**

A specific RACF user ID will be notified when access is denied. The assumption here is the user ID will monitor these notifications, that is, they will be logged on.

**An erase-on-scratch indication**

For data sets data management can physically erase contents of deleted data sets. System settings may override this setting.

**A volume and a unit (if data set)**

We assume that we are not dealing with VSAM files for these parameters. The unit can specify an installation-defined unit name, a generic device type, or a specific device address. Volume can be used for a tape data set or non-VSAM data set. You will need both if the data set is not cataloged.

**A security retention period**

This is the **RETPD** parameter and a value of 99999 means it will never expire. This applies only for tape data sets.

Locating a resource profile is shown in Figure 3-32 on page 61.

### 3.30 Data set profiles

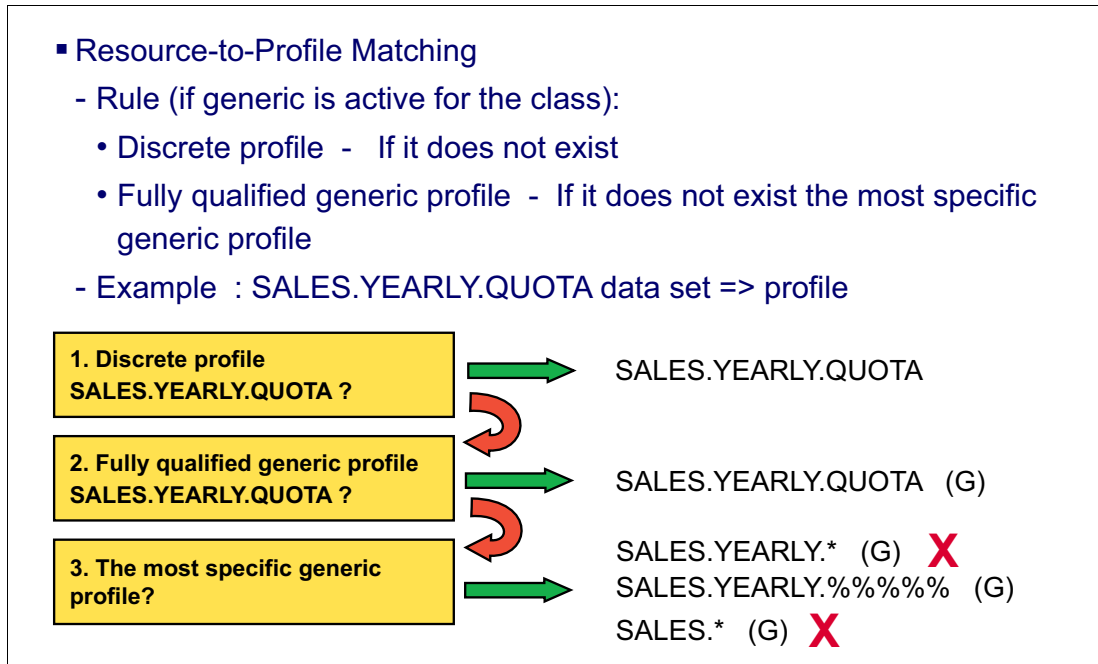


Figure 3-32 Locating a resource profile

To locate a resource profile:

- ▶ RACF looks for a discrete profile if no discrete profile is found.
- ▶ RACF looks for a generic profile and then uses the most qualified generic profile available.

However, we need to consider the impact of global access checking. A user can create or access a data set only if the data set is RACF protected by either a discrete or generic profile, or the access is allowed by global access checking (if activated for that resource class that is under scrutiny). RACF will use global access checking before other kinds of access authority checks such as security label checking or access list checking.

See *z/OS Security Server (RACF) Security Administrator's Guide*, SA23-2289, for more details about how this process works.

Some of the generic profile naming for general resources has been enhanced with some of the same concepts as generics for data set profiles as valid generic characters, as follows:

- \* You can have an asterisk (\*) within a profile name, representing one qualifier of a resource name, or specify \* in the profile name to match more than one character in the same position of the resource name.
- \*\* You can also use a double asterisk (\*\*) to represent zero or more qualifiers within a general resource generic profile or at the end of such a profile, or specify \*\* in the profile name to match more than one character in the same position of the resource name. Use of the double asterisk (\*\*) in general resource generic profiles is not controlled by the SETROPTS EGN option, which applies only to the data set profiles. EGN means enhanced generic naming.
- % Specify % for any single non-blank character (except a period) in the same position of the resource name.

## How is a generic name built?

If we have EGN in effect how does it work?

- \* This will be converted to `*.**` and has two uses. First, as at the end of a profile name to match zero or more characters, such as `SYS1.SECOND*` and second, as a qualifier to match until the end of the profile name such as `SYS1.SECOND.*`
- \*\* We use `**` as an end or middle qualifier, for example, `SYS1.SECOND.**`. This causes a match for zero or more qualifiers. `SYS1.SECOND**` is not allowed and `SYS1.**.SECOND.**` is also not allowed.

Use a RACF variable in a profile name to define one general resource profile to protect many resources with dissimilar names when no resource grouping class is available. RACF variables can be used for general resource profiles only. You cannot use them in data set profile names. A profile that contains a RACF variable in its name is considered a generic profile.

For example, suppose that you define the following profile:

```
RDEFINE RACFVARS &ABCDEFGH ADDMEM(A B)
```

In this case, profile `X.&ABCDEFGH.Z` matches both `X.AY.Z` and `X.BY.Z`.

## Choosing between discrete and generic data set profiles

Decide which type of profile to create, as follows:

### ► Generic

Choose a generic profile for the following reasons:

- If you want to protect more than one data set with the same security requirements.
- If you have a single data set that might be deleted, then re-created, and you want the protection to remain the same, you can create a fully qualified generic profile. The name of a fully qualified generic profile matches the name of the data set it protects. Unlike a discrete profile, a fully qualified generic profile is not deleted when the data set itself is deleted.

### ► Discrete

Choose a discrete profile for the following reasons:

- To protect one data set that has unique security requirements. The name of a discrete profile matches the name of the data set it protects.
- To allow changes to a data set profile to take effect immediately, without needing to refresh in-storage copies of the profile.

In Figure 3-32 on page 61, a resource manager issues a security check for the data set `SALES.YEARLY.QUOTA`. Three different types of profiles can be defined in the RACF database:

- A discrete profile
- A fully qualified generic profile
- The most specific generic profile

The example shows that RACF looks for a profile in the order shown. If no discrete profile is found, check for a fully qualified generic profile. If not found, find the most specific generic profile, which is the second one in the example, `SALES.YEARLY.%%%%`.

**Note:** By using generic profiles, your installation can reduce both the number of profiles that are required to protect data sets and the size of the RACF database, thus making RACF protection easier to administer. In addition, generic profiles are loaded into storage when first needed, are not deleted when the data set they protect is deleted, and are not volume-specific (that is, data sets protected by a generic profile can reside on any volume).

You can create a profile with a generic name when the following is true for the class of the profile:

SETROPTS GENERIC(DATASET) option is in effect.

This option allows the creation of generic profiles and also causes RACF to use generic profiles during authorization checking.

Defining data set profiles is shown in Figure 3-33.

### 3.31 Defining data set profiles

- Define a data set profile
  - ADDSD 'SALES.YEARLY.QUOTA'
- RDEFINE to add a profile for the resource
  - RDEFINE DATASET SALES.YEARLY.QUOTA UACC(NONE)
- Define who has access to data set
  - PERMIT SALES.YEARLY.QUOTA CLASS(DATASET) ID(JANE) ACCESS(READ)
  - PERMIT places specified users into an access list

Figure 3-33 Defining data set profiles

#### Defining data set profiles

Use the **ADDSD** command to add RACF protection to data sets with either discrete or generic profiles.

The **ADDSD** command adds a profile for the data set to the RACF database to control access to the data set. It also places the user ID on the access list and gives ALTER authority to the user ID unless SETROPTS NOADDCREATOR is in effect.

#### Data set profiles

By default, RACF expects a data set name (and the data set profile name) to consist of at least two qualifiers. RACF also expects the high-level qualifier of the data set profile name to be either a RACF defined user or a RACF defined group name.

Each data set profile defined to RACF requires a RACF defined user or group as the owner of the profile. The owner (if a user) has full control over the profile, including the access list.

If the owner of the data set profile is a group, users with group-SPECIAL in that group have full control over the profile.

Ownership of data set profiles is assigned when the profiles are defined to RACF. Note that ownership of a data set profile does not mean that the owner can automatically access that data set. To access a data set, the owner must still be authorized in the profile's access list, unless the high-level qualifier of the profile name is the owner's user ID.

### **Data set profile examples**

The **ADDSD** command in Figure 3-33 on page 63 specifies that no users have access to the data set except the creator of the profile because the universal access, UACC, is none.

To allow users to have access to the data set, the **PERMIT** command shown specifies that user ID JANE has only **READ** access to the data set, **ACCESS(READ)**. User ID JANE now exists in the access list for the data set profile using the **PERMIT** command.

Figure 3-34 on page 65 shows the data set profile access list.

## 3.32 Data set profile access list

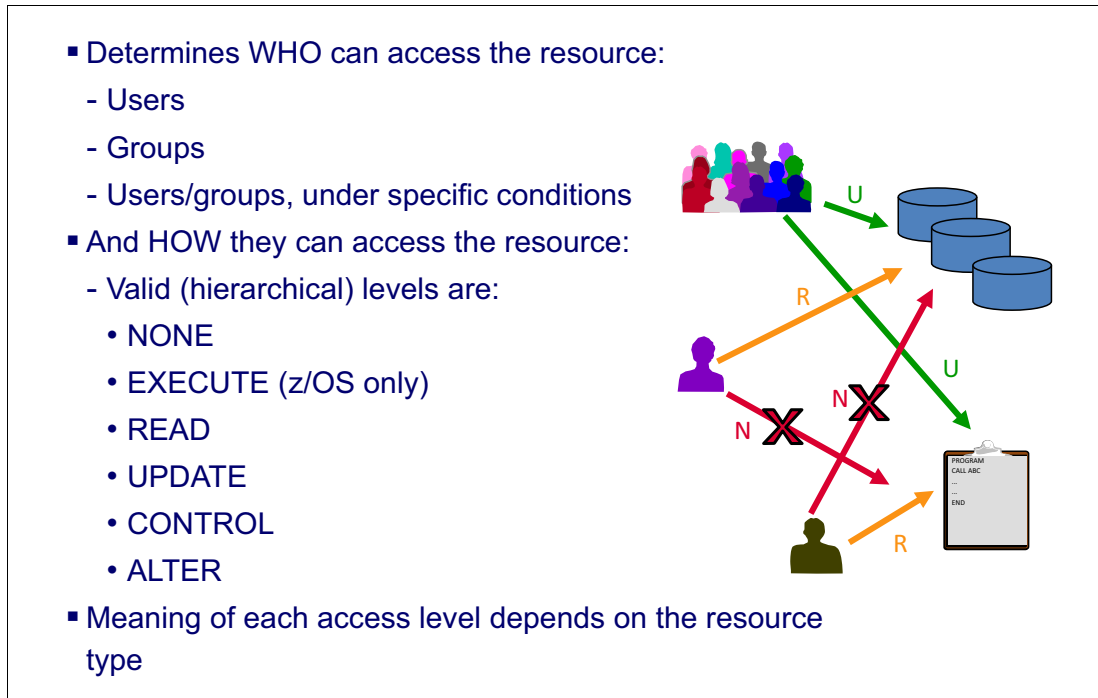


Figure 3-34 Data set profile access list

### Who can access this resource

When a user requests access to a RACF protected resource (such as a data set), the resource manager issues a RACF authorization request. SAF then passes it to RACF, who then determines if this user ID or the GROUP that it belongs to is on the access list for this resource.

How does a user ID or its group get on an access list for a resource? Normally we would use the RACF **PERMIT** command to maintain a list of users and groups authorized to access a particular resource. RACF provides two types of access lists:

**Standard** The standard access list includes the user IDs and group names authorized to access the resource and the level of access granted to each.

**Conditional** The conditional access list includes the user and group names authorized to access the resource and the level of access granted to each when a certain condition is met, such as the name of a program, or the system it is running on matches an SMF ID.

A separate **PERMIT** command is needed to establish an entry in each list.

### How can they access this resource

Types of access levels include:

**ALTER** ALTER allows users to read, update, delete, rename, move, or scratch the data set.

When specified in a discrete profile, ALTER allows users to read, alter, and delete the profile itself including the access list.

ALTER does not allow users to change the owner of the profile using the **ALTDSD** command. However, if a user with ALTER access authority to a discrete data set profile renames the data set, changing the high-level qualifier to his or her own user ID, both the data set and the profile are renamed, and the OWNER of the profile is changed to the new user ID.

When specified in a generic profile, ALTER gives users no authority over the profile itself.

<b>NONE</b>	The specified user or group is not permitted to access the resource or list the profile.
<b>EXECUTE</b>	For a private load library, EXECUTE allows users to load and execute, but not to read or copy programs (load modules) in the library.
<b>READ</b>	Allows users to access the data set for reading only. (Note that users who can read the data set can copy or print it.)
<b>UPDATE</b>	Allows users to read from, copy from, or write to the data set. UPDATE does not, however, authorize a user to delete, rename, move, or scratch the data set.
<b>CONTROL</b>	For VSAM data sets, CONTROL is equivalent to the VSAM CONTROL password; that is, it allows users to perform improved control interval processing. This is control-interval access (access to individual VSAM data blocks), and the ability to retrieve, update, insert, or delete records in the specified data set. For non-VSAM data sets, CONTROL is equivalent to UPDATE.

Figure 3-35 on page 67 shows sample output from the **LISTDSD** command.



### 3.33 Add a data set profile

- Add a data set profile: `ADDSD 'JAMES.*'`
- List the data set profile: `LISTDSD DATASET('JAMES.*')`

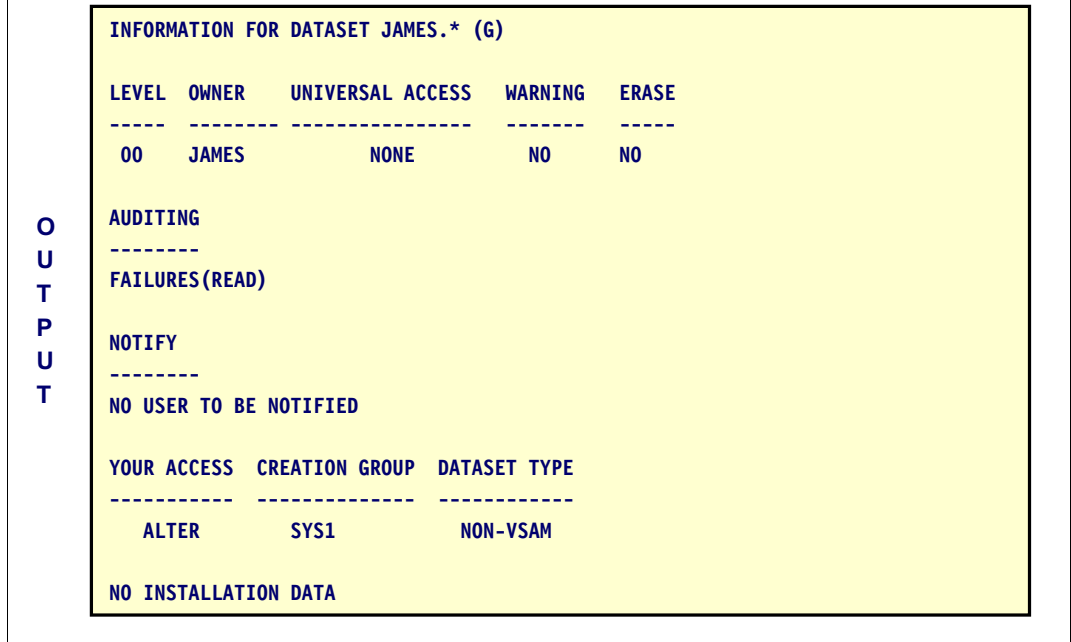


Figure 3-35 Add a data set profile

#### How to add a data set profile

When you define data set profiles to RACF, you can use either standard or nonstandard naming conventions. If you use nonstandard naming conventions, the data set naming convention table and the single-level data set names option are ways to help “fit” RACF standard naming conventions.

The descriptions of naming conventions are followed by rules for protecting and allocating user and group data sets.

By default, RACF expects a data set name (and the data set profile name) to consist of at least two qualifiers. RACF also expects the high-level qualifier of the data set profile name to be either a RACF defined user or a RACF defined group name.

This command added a generic profile for data sets with a high-level qualifier of JAMES.\*. The asterisk (\*) character is a valid generic character for more than one character in this position.

```
ADDSD 'JAMES.*'
```

Figure 3-35 shows sample output from the `LISTDSD` command.

```
LISTDSD DATASET('JAMES.*')
```

Figure 3-36 on page 68 shows a sample output from the `LISTDSD` command, which shows the auditing options as:

```
SUCCESS(UPDATE) , FAILURE(READ)
```

### 3.34 Alter a data set profile

- Alter a data set profile:
  - ALTDSD 'JAMES.\*' AUDIT(S(U),F(R))
- List the data set profile:
  - LISTDSD DATASET('JAMES.\*')

O  
U  
T  
P  
U  
T

INFORMATION FOR DATASET JAMES.* (G)				
LEVEL	OWNER	UNIVERSAL ACCESS	WARNING	ERASE
00	JAMES	NONE	NO	NO
AUDITING				
-----				
SUCCESS(UPDATE), FAILURES(READ)				
NOTIFY				
-----				
NO USER TO BE NOTIFIED				
YOUR ACCESS		CREATION GROUP	DATASET TYPE	
-----		-----	-----	
ALTER		SYS1	NON-VSAM	
NO INSTALLATION DATA				

Auditing options have changed

Figure 3-36 Alter a data set profile

#### How to alter a data set profile

Use the **ALTDSD** command to:

- ▶ Modify an existing discrete or generic data set profile.
- ▶ Protect a single volume of either a multivolume tape data set or a multivolume, non-VSAM DASD data set. At least one volume must already be RACF protected.
- ▶ Remove RACF protection from either a single volume of a multivolume tape data set or a single volume of a multivolume, non-VSAM DASD data set. You cannot delete the last volume from the profile.

Figure 3-36 shows the output for the following command to alter the auditing options for the previously created data set, JAMES.\*:

```
ALTDSD 'JAMES.*' AUDIT(S(U),F(R))
```

The command also specifies which new access attempts you want to log to the SMP data set:

- SUCCESS S(U)** Indicates that you want to log authorized accesses to UPDATE, CONTROL, and ALTER.
- FAILURES F(R)** Indicates that you want to log detected unauthorized access attempts at any level.

Figure 3-37 on page 69 shows how to list a data set profile matching a mask.

### 3.35 Search RACF database using a mask

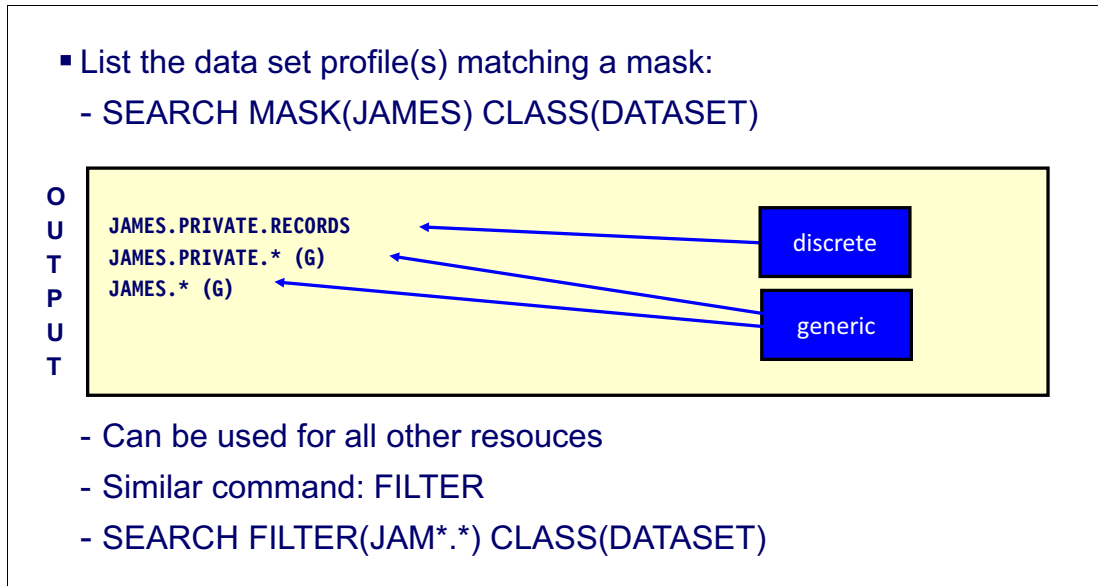


Figure 3-37 Search the RACF database

#### List a data set profile matching a mask

The **SEARCH** command obtains a list of RACF profiles, users, and groups from the RACF DATABASE using search criteria specified.

**MASK** specifies the strings of alphanumeric characters used to search the RACF database. This data defines the range of profile names selected. The two-character strings together must not exceed 44 characters for a tape or DASD data set name, or, for general resource classes, the length specified in the class descriptor table.

The visual shows a **SEARCH** command with the search criteria, **MASK**.

```
SEARCH MASK(JAMES) CLASS(DATASET)
```

This command allows RACF to list profiles starting with the **MASK**, in this case **JAMES**.

A second example allows RACF to list all profiles containing the filter string.

```
SEARCH FILTER(JAM*.* ) CLASS(DATASET)
```

Figure 3-38 on page 70 shows how to list a cataloged data set.

### 3.36 Data set-related commands

- List the cataloged data set(s) protected by a profile:  
- LISTDSD DATASET('JAMES.\*') DSNS

**O  
U  
T  
P  
U  
T**

```
INFORMATION FOR DATASET JAMES.* (G)

LEVEL  OWNER  UNIVERSAL ACCESS  WARNING  ERASE
-----  -----  -----  -----  -----
  00    JAMES          NONE          NO        NO

AUDITING
-----
SUCCESS(UPDATE) , FAILURES(READ)
...

CATALOGUED DATA SETS AFFECTED BY PROFILE CHANGE
-----
JAMES.PGMLIB
JAMES.WORK.EXEC
```

Figure 3-38 List a data set

#### List a cataloged data set

Figure 3-38 shows sample output from the following **LISTDSD** command, which allows RACF to list data sets protected by a profile (in this case, the JAMES.\* data set profile):

```
LISTDSD DATASET('JAMES.*') DSNS
```

The parameter **DSNS** specifies that you want to list the cataloged data sets protected by the profile specified in the **DATASET** parameter. This is a usual command to show all the cataloged data sets for the profile.

Figure 3-39 on page 71 lists who has access to a data set.

### 3.37 Data set-related commands, continued

- Allow access to a data set profile:
  - PERMIT 'JAMES.\*' ID(BILL,DESIGN) ACCESS(UPDATE)
  - PERMIT 'JAMES.\*' ID(PAT) ACCESS(READ)
- List the data set profile access list:
  - LISTDSD DATASET('JAMES.\*') AUTHUSER

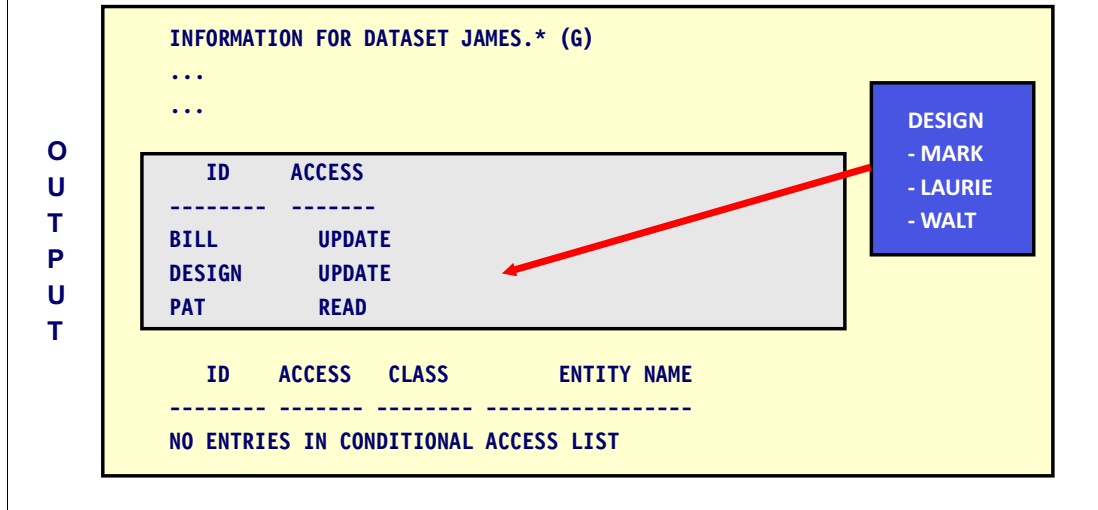


Figure 3-39 List who has access to a data set

#### List who has access to a data set profile

Figure 3-39 shows sample output from the following **PERMIT** command:

```
PERMIT 'JAMES.*' ID(BILL,DESIGN) ACCESS(UPDATE)
```

This command allows user ID Bill and the DESIGN group update access to the files protected by the James.\* data set profile. Mark, Laurie, and Walt part of the DESIGN group will have UPDATE access, unless the access list contains their user ID with another level of access.

```
PERMIT 'JAMES.*' ID(PAT) ACCESS(READ)
```

User ID Pat now has read access to the files that are protected by the JAMES.\* profile. In Figure 3-39, we see the **AUTHUSER** parameter on the **LISTDSD** command. This provides additional information including access statistics (if any) on the output.

Figure 3-40 on page 72 shows how to add a general resource profile.

### 3.38 General resources-related commands

- Add a general resource profile:
  - RDEF PROGRAM MYMUSIC  
ADDMEM('JAMES.PGMLIB'/VOL123/NOPADCHK)
- List the resource profile: RL PROGRAM MYMUSIC

CLASS	NAME			
PROGRAM	MYMUSIC			
MEMBER CLASS NAME				
-----				
PMBR				
DATA SET NAME				
			VOLSER	PADS CHECKING
-----			-----	-----
JAMES.PGMLIB			VOL123	NO
LEVEL OWNER UNIVERSAL ACCESS YOUR ACCESS WARNING				
-----				
00	JAMES	NONE	NO	NO
INSTALLATION DATA				
-----				
NONE				

Figure 3-40 Add a general resource profile

#### How to add a general resource profile

Figure 3-40 shows sample output after the following **RDEFINE (RDEF)** command, which defines a new resource profile called MYMUSIC that will run in PROGRAM class:

```
RDEF PROGRAM MYMUSIC ADDMEM('JAMES.PGMLIB'/VOL123/NOPADCHK)
```

The program MYMUSIC is in JAMES.PGMLIB member on DASD volume VOL123.

Setting NOPADCHK means that RACF will not check for program-accessed data sets when a user is executing the control programs. We can do this by defining entire libraries of modules (such as ISPF) as controlled programs without then having to grant each of these modules access to many program-accessed data sets.

The command **RLIST (RL)** in Figure 3-40 is used to display information of this resource profile.

Figure 3-41 on page 73 shows how to change universal access authority.

### 3.39 General resources-related commands, continued

- Alter a general resource profile:
  - RALT PROGRAM MYMUSIC UACC(READ)
- List the data set profile: RL PROGRAM MYMUSIC

```

O
U
T
P
U
T

CLASS      NAME
-----
PROGRAM    MYMUSIC

MEMBER CLASS NAME
-----
PMBR

DATA SET NAME                                VOLSER    PADS CHECKING
-----
JAMES.PGMLIB                                VOL123     NO

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----
00    JAMES      READ              READ         NO

INSTALLATION DATA
-----
NONE

```

Figure 3-41 Change universal access authority

#### How to change universal access authority

Figure 3-41 shows sample output after the following **RALTER (RALT)** command, where it specifies the PROGRAM as the class name.

```
RALT PROGRAM MYMUSIC UACC(READ)
```

The universal access authority (UACC) is the default access to a resource if the user or group is not specifically permitted access to the resource. The **RALTER** command has set the default access to MYMUSIC to READ.

The command **RLIST (RL)** shows the resource profile for MYMUSIC where its class is PROGRAM.

Figure 3-42 on page 74 shows how to control access to a resource profile.

### 3.40 General resources-related commands, continued

- Control access to a general resource profile:
  - PERMIT MYMUSIC CLASS(PROGRAM) ID(MARVIN) ACCESS(NONE)
- List the data set profile access list:
  - RL PROGRAM MYMUSIC AUTHUSER

O  
U  
T  
P  
U  
T

CLASS	NAME
-----	----
PROGRAM	MYMUSIC
MEMBER CLASS NAME	
-----	
PMBR	
...	
...	

USER	ACCESS	ACCESS COUNT
-----	-----	-----
MARVIN	NONE	000000

ID	ACCESS	CLASS	ENTITY NAME
-----			
NO ENTRIES IN CONDITIONAL ACCESS LIST			

Specifically Disallow Access

O  
U  
T  
P  
U  
T

Figure 3-42 Permit access to a resource

#### How to control access to a resource profile

Figure 3-42 shows sample output after the **PERMIT (PE)** command:

```
PERMIT MYMUSIC CLASS(PROGRAM) ID(MARVIN) ACCESS(NONE)
```

Despite the UACC(READ) on the resource profile, MARVIN cannot access the resource because NONE is specified in the access list. MARVIN identifies the user ID or group name, who are RACF defined users or groups and whose authority to access the resource you are removing.

Figure 3-43 on page 75 shows the **SETROPTS** command.



## 3.41 SET RACF system options

- SETROPTS - set system-wide RACF options
  - PASSWORD rules: syntax, historic, number of attempts, etc.
  - CLASSACT: activate new classes
  - RACLIST(classname) REFRESH: update in-storage information
- Authorization required

**Shareable among systems**

Figure 3-43 3.41, “SET RACF system options” on page 75

### SETROPTS command

RACF provides many system-wide options for controlling the way it works on your system. You specify most of these options by issuing the **SETROPTS** command with the appropriate operands. One example is to set the system-wide valid password interval:

```
SETROPTS PASSWORD INTERVAL(30)
```

The INTERVAL suboperand specifies the system default for the number of days that the user's password is to remain valid; this is the maximum change interval. The example specifies that each user's password remains valid for 30 days. This is an example how an enterprise-wide policy on password expiry can be implemented.

### CLASSACT parameter

When you install a new RACF system, initially only a few RACF classes are active (for example USER, GROUP, and DATASET); other classes (for example TAPEVOL and TSOPROC) are inactive. For example, if you want your tape volumes to be protected by RACF, you have to *activate* the TAPEVOL class using the following command:

```
SETROPTS CLASSACT(TAPEVOL)
```

The classes that CLASSACT specifies must already be defined by entries in the class descriptor table for which RACF protection is to be in effect. So, TAPEVOL must already be in the CDT.

### RACLIST REFRESH parameter

The system options are stored in the RACF database and if your installation has activated SETROPTS RACLIST processing for a particular resource class, the information is stored in *in-storage profiles* too. Using the **SETROPTS** command with the **REFRESH** parameter allows these profiles to be updated dynamically.

The following example updates the profile in the class TSOPROC dynamically:

```
SETROPTS RACLIST(TSOPROC) REFRESH
```

## Authorization

The **SETROPTS** command is very powerful and, therefore, most of the options require the SPECIAL attribute.

**Note:** For further information about the required authority, refer to *z/OS Security Server RACF Command Language Reference, SA23-2292*. The description for each RACF command contains a heading called *Authorization required*.

The following pages show some examples (but not all) of system-wide settings. They are grouped to:

- ▶ STATISTIC-related options
- ▶ PASSWORD options
- ▶ Data set related options
- ▶ CLASS-related options
- ▶ AUTHORIZATION Checking options
- ▶ TAPE-related options
- ▶ Other initial setup-related options

Figure 3-44 shows statistic-related options.

## 3.42 Statistic-related options

- Activating statistics Collection (STATISTICS)
- Activating statistics for user verification (INITSTATS)
- Revoking unused user IDs (INACTIVE)

Figure 3-44 3.42, "Statistic-related options" on page 76

An installation can record two types of RACF statistics:

- ▶ STATISTICS, this is a parameter of the **SETROPTS** command. Using this causes RACF to record or not record statistical information for discrete profiles within the specified class name. If you specify a \* value, the valid class names are DATASET and those classes defined in the class descriptor table.
- ▶ INITSTATS, which allows statistics to be recorded during the user verification such as date and time.

### Activating statistics collection (STATISTICS option)

For some reasons (for example, if a specific resource has unique security concerns and, therefore, is protected by a discrete profile), it might be useful to have statistical data about a resource concerning *how* that resource is being accessed and *how many times* it is being accessed. The SETROPTS STATISTICS option provides this information. RACF maintains two sets of statistics in a discrete resource profile. One set counts all activity for the resource or profile. The other set counts activity for each entry in the access list. The following command turns STATISTICS on for the resources in the class TSOPROC:

```
SETROPTS STATISTICS(TSOPROC)
```

**Attention:** Remember that the initiation of STATISTICS is *system-wide* for all discrete profiles within a particular resource class across your system. Depending on the number of discrete profiles in the various resource classes, turning on STATISTICS can negatively affect performance.

When a new RACF database is initialized, the default is STATISTICS off (NOSTATISTICS) for all classes.

**Tip:** It is recommended that you keep STATISTICS off until your installation has had an opportunity to evaluate the need for STATISTICS versus the potential impact on performance.

For details, see *z/OS Security Server RACF System Programmer's Guide*, SA23-2287.

### Activating statistics for user verification (INITSTATS option)

When a new RACF database is initialized, the default is INITSTATS *on*. INITSTATS records statistics on all user profiles in the system.

**Note:** Although INITSTATS affects performance because of I/O to the database, it is recommended that INITSTATS stays on, because it allows you to use other options to provide additional security at logon.

INITSTATS is required if your installation wants to take advantage of the following options:

- ▶ SETROPTS INACTIVE option
- ▶ SETROPTS PASSWORD option with parameter REVOKE, HISTORY, and WARNING

### Revoking unused user IDs (INACTIVE option)

The INACTIVE operand of the **SETROPTS** command causes RACF to revoke the user's right to use the system if the user ID has remained unused beyond a specified number of days. The following command causes RACF to revoke a user ID if it is unused for over 30 days:

```
SETROPTS INACTIVE(30)
```

If you issue the SETROPTS INACTIVE(30) command and if a user has not done any of the following activities in 31 days, that user is considered revoked:

- ▶ Logged on
- ▶ Submitted a job
- ▶ Changed the user's password by any method
- ▶ Attempted an unsuccessful logon
- ▶ Received a directed command or output from RACF

The INACTIVE option applies also to new RACF defined user IDs if the new user ID is not used within the number of days specified by SETROPTS INACTIVE.

Figure 3-45 on page 78 shows password-related options.

**Note:** The user is not actually revoked. RACF revokes the user the next time the user attempts to enter the system.

## 3.43 Password-related options

- SETROPTS PASSWORD
  - Allowing mixed-case passwords
  - Establishing syntax rules
  - Setting the maximum and minimum change interval
  - Extending password and user ID processing
    - warning in relation to change interval
    - password and password phrase history
    - revoking user IDs using consecutive incorrect passwords or password phrases

Figure 3-45 3.43, “Password-related options” on page 78

The examples in this section show some of the SETROPTS PASSWORD suboperand, which gives you the possibility to specify system-wide options regarding passwords. An optional password phrase can be used. Most of the information for the password also applies to password phrases.

### Allowing mixed-case passwords

By default, NOMIXEDCASE is in effect and mixed-case passwords are not supported. Mixed-case passwords are more secure and harder to guess than uppercase passwords. You can allow mixed-case passwords with the following command.

```
SETROPTS PASSWORD(MIXEDCASE)
```

#### Attention:

- ▶ If you want to allow mixed-case passwords, be sure that mixed-case content is permitted by your password syntax rules.
- ▶ z/OS 1.7 is the first release that supports mixed-case passwords. If you share the RACF database with earlier systems that do not support mixed-case RACF passwords or if you use a mix of applications that do and do not support mixed-case passwords, do not activate the SETROPTS PASSWORD(MIXEDCASE) option.

### Establishing syntax rules

You can establish up to eight password syntax rules to verify that new passwords meet the installation standards. These rules allow you to control:

- ▶ The minimum and maximum length of passwords
- ▶ The character content of installation-selected positions in the passwords

You establish these rules by using the RULE $n$  suboperand specified by the PASSWORD operand of the SETROPTS command. The following example shows how you can establish a syntax rule for new passwords for your installation:

```
SETROPTS PASSWORD(RULE1(LENGTH(8) VOWEL(1,3,5:8) NUMERIC(2,4)))
```

The command establishes syntax rule RULE1. Syntax rule RULE1 specifies that new passwords must be eight characters in length, must contain vowels in positions 1, 3, 5, 6, 7, and 8, and must contain numbers in positions 2 and 4. Thus, the password A2E9DEOM follows the rule, and C3DFFER5 does not.

The syntax rules for password phrases are “hardcoded” but can be controlled by use of an exit such as ICHPWX11.

**Note:** Your changes take effect for current users only when they change their passwords. For new users, the changes take effect when the new user logs on for the first time.

### Setting the maximum and minimum change interval

The INTERVAL suboperand specifies the system default for the maximum number of days that a user’s password is to remain valid. The MINCHANGE suboperand specifies the system default for the minimum number of days that must pass between a user’s password changes. To specify that each user’s password remains valid for 45 days and that no user can change passwords more often than every seven days, use the following command:

```
SETROPTS PASSWORD(INTERVAL(45) MINCHANGE(7))
```

**Note:** z/OS 1.7 is the first release that supports MINCHANGE. The installation default is zero (0) days for minimum change interval. The value MINCHANGE(0) allows users to change passwords more than once each day.

### Extending password and user ID processing

The WARNING suboperand specifies when RACF issues a password expiration message each time a user logs on to TSO or submits a batch job with a password within a specified number of days (in the following example, five days) before the password expires:

```
SETROPTS PASSWORD(WARNING(5))
```

The HISTORY suboperand specifies the number of previous passwords (in the following example, 10) that RACF saves and compares with an intended new password:

```
SETROPTS PASSWORD(HISTORY(10))
```

REVOKE specifies how many consecutive password verification attempts RACF permits before it revokes a user ID on the next attempt:

```
SETROPTS PASSWORD(REVOKE(3))
```

**Note:** Option INITSTATS is prerequisite of the options WARNING, HISTORY, and REVOKE. It requires this in order to collect statistics so it has data to base its logic on.

Figure 3-46 on page 80 shows data set-related options.

## 3.44 Data set-related options

- Activating Enhanced Generic Naming for the DATASET Class (EGN)
- RACF-Protecting All Data Sets (PROTECTALL)
- Bypassing Automatic Data Set Protection (NOADSP)
- Preventing Access to Uncataloged Data Sets (CATDSNS)
- Displaying and Logging Real Data Set Names (REALDSN)
- Protecting Data Sets with Single-Qualifier Names (PREFIX)
- Erasing Scratched or Released DASD Data (ERASE)
- Protecting DFP-Managed Temporary Data Sets (TEMPDSN)
- Authorizing access to z/OS UNIX files and directories (FSACCESS)

Figure 3-46 3.44, “Data set-related options” on page 80

### Activating enhanced generic naming for the DATASET class

When you first initialize the RACF database, enhanced generic naming is not in effect (NOEGN). Use the following command so that RACF allows you to specify the generic character, double asterisks (\*\*), in addition to the generic characters, asterisk (\*), and percentage (%).

```
SETR_OPTS EGN
```

**Note:** IBM strongly recommends that you do *not* deactivate enhanced generic naming after data set profiles have been created while enhanced generic naming was active.

### RACF protecting all data sets (PROTECTALL option)

If PROTECTALL is active, a user can create or access a data set only if the data set is RACF protected. Use the following command to activate this option:

```
SETR_OPTS PROTECTALL
```

**Note:** Before activating this option, activate generic profile checking also for the DATASET class as shown in the following command:

```
SETR_OPTS GENERIC(DATASET)
```

PROTECTALL also has a warning option that allows the request even though the data set is not protected but sends a warning message to the user and the z/OS console. For example:

```
SETR_OPTS PROTECTALL(WARNING)
```

For further considerations on the PROTECTALL option, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

## Bypassing automatic data set protection (NOADSP option)

With the installation default ADSP operand in effect, RACF creates discrete data set profiles automatically when users who have the ADSP attribute create new data sets.

**Note:** We recommend the NOADSP option because it reduces the number of data set profiles in the RACF database. Using generic data set profiles is generally more efficient.

You can change the installation default using the following command:

```
SETROPTS NOADSP
```

## Preventing access to uncataloged data sets (CATDSNS option)

You can use the CATDSNS operand of the **SETROPTS** command to keep users who do not have the SPECIAL attribute from gaining access to data sets that Data Facility Product (DFP) controls. These data sets include system temporary data sets and data sets that are not cataloged. With this in place, users cannot read data sets from tape, and they cannot read from or write to DASD data sets.

**Note:** Because of the big impact this option can have on data processing, it might be reasonable to specify CATDSNS(WARNING) before you plan to activate it in failure mode.

## Displaying and logging real data set names (REALDSN option)

Putting the REALDSN option into effect ensures that log printouts and operator messages identify data sets by their real names rather than by the data set names that are created by installation exit routines to conform to RACF naming conventions.

## Protecting data sets with single-qualifier names (PREFIX option)

If your installation has data sets names consisting of only a single qualifier (that is, single-level names) and if you want RACF to protect this data set, you must specify the PREFIX option:

```
SETROPTS PREFIX(myh1q)
```

RACF internally modifies single-qualifier names by adding the high-level qualifier (in this case myh1q) when it processes requests for the data set. The prefix must be an existing group name and cannot be the name used as the high-level qualifier of any actual data sets or data set profiles.

## Erasing scratched or released DASD data (ERASE option)

If erase-on-scratch processing is active and a DASD data set profile has the erase indicator set, ERASE specifies that data management is to erase the contents of any scratched or released data set extents that are part of a DASD data set protected by that profile.

You can use the SECLEVEL suboperand to control this erase process further. This tells the data management to erase all scratched data sets that have a security level equal to or greater than the security level that you have, albeit indirectly. SECLEVEL suboperand is a name and that name must be in the SECLEVEL profile in the SECDATA class. This is where the actual security level number is obtained.

The ERASE option applies to DASD data sets only, not *tape* data sets, unless you set the TAPEAUTHDSN option in the DEVSUPxx member of SYS1.PARMLIB. See “Erasing Scratched or Release Data (ERASE Option)” in *z/OS Security Server RACF Security’s Administrator’s Guide*, SA23-2289, for more information.

## Protecting DFP-managed temporary data sets

You can protect DFP-managed temporary data sets. Normally, these data sets are considered protected from any accesses except by the job or session that created them and, therefore, do not need to be protected by RACF. However, the following situations can leave a temporary data set unprotected:

- ▶ A system failure
- ▶ An initiator failure or initiator termination by the FORCE command
- ▶ An automatic restart—between the failure and the restart

In these cases, if the TEMPDSN class is active, only users with the OPERATIONS attribute can scratch any residual DFP-managed temporary data sets remaining on a volume.

**Note:** The user with the OPERATIONS attribute can access the data set only to scratch the data set. No other access is allowed (such as would be allowed by READ or UPDATE access authority to the data set).

To activate the TEMPDSN class, enter:

```
SETROPTS CLASSACT(TEMPDSN)
```

**Important:** Plan carefully when to activate the TEMPDSN class to avoid a situation where current users or jobs are using temporary data sets. Otherwise, you might cause users or jobs to receive an ABEND.

When you share the RACF database with a downlevel system running z/OS V1R12 or earlier, avoid activating the TEMPDSN class when current users or jobs are using temporary data sets. It might cause users or jobs on the downlevel system to receive an ABEND as their current access is removed.

## Authorizing access to z/OS UNIX files and directories

To place the access of z/OS UNIX zFS file system objects within the scope of the RACF security administrator, an optionally enforced access control check will be implemented that validates a user's authority to access the file system objects using a RACF FSACCESS class profile. This profile is defined as the z/OS UNIX file system name.

This access check will be based solely on the user's z/OS user ID, meaning that superuser authority will not be used or influence the outcome of this access control check. This check is intended to be coarse grained, in that if the user is not authorized to this profile, no further checking will be performed. If the user is authorized to the z/OS UNIX zFS file system container profile, the file permission bits and access control lists (ACLs) that are associated with the individual z/OS UNIX file system objects will then govern the access to the file or directory, as it is done today.

This coarse grained access check provides an easy means to prove during compliance audits that a user does not have access to sensitive data that may reside within the z/OS UNIX zFS file systems.

The following steps restrict access to a zFS file system:

1. Define a profile in the FSACCESS class to protect each zFS file system. The profile name is the name of the MVS data set that contains the file system:

```
RDEFINE FSACCESS OMVS.ZFS.WEBSRV.TOOLS UACC(NONE)
```



2. If multiple file systems are stored in data sets with similar names, you can define a generic profile name to protect multiple file systems. Before you define a generic profile in the FSACCESS class, enable generics for the class, as follows:

```
SETROPTS GENERIC(FSACCESS)
RDEFINE FSACCESS OMVS.ZFS.WEBSRV.** UACC(NONE)
```

3. Authorize selected users and groups with UPDATE access:

```
PERMIT OMVS.ZFS.WEBSRV.TOOLS CLASS(FSACCESS) ID(GROUPB USER19) ACCESS(UPDATE)
```

4. Activate your profile changes in the FSACCESS class, as follows. If the FSACCESS class is not already active, activate and RACLIST it. If active, replace RACLIST parameter with the REFRESH parameter:

```
SETROPTS CLASSACT(FSACCESS) RACLIST(FSACCESS)
```

For a more detailed explanation, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

Figure 3-47 shows class-related options.

## 3.45 Class-related options

- Activating general resource classes (CLASSACT)
- Activating generic profile checking and generic command (GENERIC and GENCMD)
- Processing activating global access checking facility (GLOBAL)
- Activate in-storage profile processing (RACLIST and GENLIST)
- Refreshing in-storage profiles (REFRESH)
- Restricting the creation of general resource profiles (GENERICOWNER)
- Automatic omission of creator's user ID from access list (NOADDCREATOR)

Figure 3-47 3.45, "Class-related options" on page 83

### Activating general resource classes (CLASSACT)

The system-wide security administrator specifies in which general resource classes RACF provides access authorization checking. You can specify this option for selected general resource classes with the CLASSACT operand of the SETROPTS command. The following example shows how to specify RACF access authorization checking for the TERMINAL and CONSOLE resource classes:

```
SETROPTS CLASSACT(TERMINAL CONSOLE)
```

**Important:**

- ▶ We do not recommend that you activate *all* RACF classes. Activate only those classes that are important to your installation because some classes have a default return code of eight. Activate those classes only after you define the necessary profiles to allow access to resources, using the following command:

```
SETROPTS NOCLASSACT(TERMINAL)
```

This NOCLASSACT operand indicates that RACF performs no access authorization checking for the specified general resource classes.

- ▶ If you activate a class using SETROPTS CLASSACT, RACF activates all classes in the class descriptor table that have the same position ID (POSIT) value as the class that you specify. The same effect is true for the other class-related options. Therefore, we do not mention this note in every topic. For details, see *z/OS Security Server RACF Command Language Reference*, SA23-2292.

## Activating generic profile checking and generic command processing (GENERIC and GENCMD)

You can activate or deactivate *generic profile checking* and *generic command processing* on a class-by-class basis. The following example shows how to activate generic profile checking and generic command processing for the DATASET class:

```
SETROPTS GENERIC(DATASET)
```

Generic profile command processing is activated automatically for all classes for which generic profile checking is activated. It is strange, but the SETROPTS GENERIC command implicitly turns on GENCMD for the stated class.

NOGENERIC and NOGENCMD are in effect when a RACF database is first initialized using IRRMIN00.

**Tip:** We recommend that you use generic profiles, if possible, to protect multiple resources and, thus, to ease the administration. Consider issuing SETROPTS GENERIC(\*) so that generic profiles and generic command processing are usable in all classes.

If you want to perform maintenance on the generic profiles in the RACF database, you might want to temporarily deactivate generic profile checking but allow RACF command processors to update generic profiles. You can specify this environment with the NOGENERIC and GENCMD operands of the SETROPTS command. The following example shows how to specify this environment for the DATASET class.

```
SETROPTS NOGENERIC(DATASET) GENCMD(DATASET)
```

For more information about this topic, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

## Activating global access checking facility (GLOBAL)

RACF provides *global access checking* to improve performance of RACF authorization checking for selected resources. You can use global access checking for public resources that are accessed frequently. The global access checking table is maintained in storage and is checked early in the RACF authorization checking sequence. If an entry in the global access checking table allows the requested access to a resource, RACF performs no further authorization checking:

```
SETROPTS GLOBAL(classname)
```

**Attention:**

- ▶ Because RACF performs global access checking *before* many of the other kinds of access authority checks, such as security label checking or access list checking, global access checking might allow access to a resource you are otherwise protecting. To avoid a security exposure to a sensitive resource, do not create an entry in the global access checking table for a resource that is protected by a profile containing a security level, security category, or security label.
- ▶ When global access checking allows a request, RACF performs no logging other than that requested by the SETROPTS LOGOPTIONS command. See also “LOGOPTIONS: Activating auditing for access attempts by class” on page 95.

For further consideration before activation of global access checking, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

**Activate in-storage profile processing (RACLIST and GENLIST)**

In-storage profiles can help the administrator maximize performance of the RACF database. RACF provides processing to activate in-storage profiles both generic and discrete, for the classes specified. The SETROPTS operands are **GENLIST** and **RACLIST**.

**Note:** RACF does not allow you to specify SETROPTS GENLIST and SETROPTS RACLIST for the same general resource class at the same time.

The RACLIST operand on the SETROPTS command copies the base segments of generic and discrete profiles from the RACF database into storage. The profile copies are put in their own data space.

The GENLIST operand on the SETROPTS command improves performance by copying generic profiles from the RACF database. The profile copies are put in an extended common storage area (ECSA).

For more information about when to use RACLIST and GENLIST processing, see *z/OS Security Server RACF System Programmer's Guide*, SA23-2287. Classes for which RACLIST processing is recommended are listed there.

**Note:** A general resource class must be active before you can activate SETROPTS GENLIST or SETROPTS RACLIST processing for that class.

**Refreshing in-storage profiles (REFRESH)**

If your installation maintains in-storage copies of resource profiles through the SETROPTS RACLIST or SETROPTS GENLIST command, changes to those profiles do not take effect on the system until a SETROPTS RACLIST REFRESH or SETROPTS GENERIC REFRESH command is issued. For details, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

To activate refreshing of SETROPTS RACLIST processing for the TSOPROC and TSOAUTH classes, use this command:

```
SETROPTS RACLIST(TSOPROC TSOAUTH) REFRESH
```

**Restricting the creation of general resource profiles (GENERICOWNER)**

RACF provides the possibility to restrict the creation of profiles in general resource classes. It prevents the creation of a more specific profile than an existing profile. You must issue the **SETROPTS GENERICOWNER** command and define a double asterisk (\*\*) profile for the class with yourself as the owner.

**Note:** The GENERICOWNER operand does not affect the DATASET class. It cannot be activated for individual classes. When active, GENERICOWNER affects all general resource classes except the PROGRAM class and general resource grouping classes.

### Automatic omission of creator's user ID from access list (NOADDCREATOR)

The SETROPTS options ADDCREATOR and NOADDCREATOR allow you to specify whether the user ID of the person who defines a resource profile is placed on the access list for that resource automatically with ALTER authority. The following command causes RACF not to place the profile creator's user ID on the profile access list:

```
SETROPTS NOADDCREATOR
```

**Note:** We recommend that you use the NOADDCREATOR option. If the creating user needs access to the profile being defined, access to the profile should be done separately, and if possible, by specifying a group and not an individual user ID. This will not apply for the DATASET and TAPEVOL classes created through RACROUTE REQUEST=DEFINE. The user ID of any profile creator is placed on the new profile's access list with ALTER authority.

Figure 3-48 shows authorization checking-related options.

## 3.46 Authorization checking-related options

- Activating list-of-groups checking (GRPLIST)
- Activating program control (WHEN(PROGRAM))
- Activating terminal control (TERMINAL(READ/NONE))

Figure 3-48 3.46, "Authorization checking-related options" on page 86

### Activating list-of-groups checking (GRPLIST)

A RACF defined user can be a member of different RACF groups. If list-of-groups checking is activated, a user's authority to access or define a resource is not based only on the authority of the user's current logon group. Access is based on the authority of any group to which the user is connected.

**Note:** If list-of-groups checking is activated and if a user is in more than one group and tries to access a resource, RACF uses the *highest* authority that is allowed by the user's list of groups and the resource's access list.

For example, if list-of-groups processing (SETROPTS GRPLIST option) is active, and user JAMES is connected to groups MFG and DESIGN, and MFG group is permitted access to FILE1.ACCOUNTS with ACCESS(NONE), DESIGN group is permitted with ACCESS(READ), and JAMES is not explicitly permitted, then JAMES's effective access to FILE1.ACCOUNTS is READ.

NOGRPLIST is in effect when RACF is using a newly initialized database. You can change this option using the following command:

```
SETOPTS GRPLIST
```

**Tip:** We recommend that you use the GRPLIST option because it eases administration and minimizes the number of times the user might have to log off and log back on to access resources.

### Activating program control (WHEN(PROGRAM))

The general resource class PROGRAM can be used with program control. Program control is activated using SETOPTS:

```
SETOPTS WHEN(PROGRAM)
```

When program control is active, RACF provides access control to load modules, and program access to data sets and SERVAUTH resources.

Access control to load modules allows only authorized users to load and execute specified load modules (programs). RACF uses profiles in the PROGRAM general resource class to control access to programs. By protecting load modules, the installation can establish controls over who can run certain programs. A program protected by a profile in the PROGRAM class is called a *controlled program*.

*Program access to data sets* allows an authorized user or group of users to access specified data sets with the user's authority to execute a certain program. That is, some users can access specified data sets at a specified access level only while executing a certain program.

*Program access to SERVAUTH class resources* allows an authorized user or a group of users to access certain IP addresses with the user's authority to execute a certain program. That is, some users can access specified IP addresses at a specified access level only while executing a certain program.

NOWHEN(PROGRAM) is in effect when a RACF database is first initialized using IRRMIN00.

**Note:** We recommend that you implement the general resource class PROGRAM from a security point of view. There are many system programmer-related programs, for example *AMASPZAP* or some *RACF utilities*, which should not be used by unauthorized users.

For details about program control, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

### Activating terminal control (TERMINAL(READ/NONE))

RACF provides the general resource class TERMINAL to control the use of terminals. The system-wide option TERMINAL(READ) or TERMINAL(NONE) is used to set the *universal access authority (UACC)* associated with undefined terminals.

The following command sets the TERMINAL class of resource in RACF to an active, system-wide status:

```
SETOPTS CLASSACT(TERMINAL) TERMINAL(READ)
```

All subsystems that use RACF to control access to terminals now have terminal checking active when this command is issued. The READ option of the TERMINAL operand indicates how RACF is to view terminals that are not defined to RACF. READ indicates that if RACF cannot find a profile for that terminal, access to the terminal is to be allowed.

To prevent undefined terminals from being used for logging on, use the following command:

```
SETROPTS TERMINAL(NONE)
```

**Attention:** Before you specify NONE, be sure that you define some terminals to RACF and give the appropriate users and groups proper authorization to use them. Otherwise, *no one can log on to your system.*

If your installation uses dynamic IP addresses instead of static VTAM defined terminal names, it is not easy to administer profiles in the RACF class TERMINAL.

Figure 3-49 shows tape-related options.

## 3.47 Tape-related options

- Activating tape data set protection (TAPEDSN)
- Activating tape volume protection (TAPEVOL )
- Establishing a security retention period for tape data sets (RETPD)

Figure 3-49 3.47, "Tape-related options" on page 88

RACF allows you to establish access requirements for both tape data sets and tape volumes.

### Activating tape data set protection (TAPEDSN)

RACF provides tape data set protection if you use the TAPEDSN operand of the SETROPTS command. When you activate tape data set protection, RACF refers to profiles in the DATASET class when verifying a user's access authority to a tape data set. The following example shows how to specify this option:

```
SETROPTS TAPEDSN
```

NOTAPEDSN is in effect when a RACF database is first initialized using IRRMIN00. In this case, RACF cannot protect individual tape data sets, although it can protect tape volumes.

### Activating tape volume protection (TAPEVOL)

You can activate tape volume protection using the CLASSACT(TAPEVOL) operand of the SETROPTS command. When you activate tape volume protection, RACF refers to profiles in the TAPEVOL class when verifying a user's access authority to a tape volume.

If both the TAPEVOL class and TAPEDSN are active, RACF maintains profiles in both the TAPEVOL and DATASET classes. Data fields within these two profiles (data set name in the TAPEVOL profile and volume serial in a discrete data set profile) link the two profiles to each other. The following example shows how to activate tape volume protection:

```
SETROPTS CLASSACT(TAPEVOL)
```

**Note:** If your installation has a tape management system, you might consider running with TAPEDSN active and TAPEVOL inactive. In this case, your tape management system, not RACF, maintains tape volume security and controls access to tape volumes.

For more information, see “Choosing Which Tape-Related Options to Use” in *z/OS Security Server RACF Security’s Administrator’s Guide*, SA23-2289.

### **Establishing a security retention period for tape data sets (RETPD)**

The RACF security retention period is the number of days that RACF protection remains in effect for a tape data set. RACF stores the value in the tape data set profile. If you specify RETPD, you must also activate TAPEDSN. The following example shows how to specify a RACF security retention period of 365 days:

```
SETROPTS RETPD(365)
```

The RACF security retention period is stored in the data set profile (specified using the RETPD operand on the **ADDSD** or **ALTDSD** command). If the data set profile does not contain a security retention period, look at one of the following:

1. For discrete profiles, RACF uses the creation date stored in the TVTOC and the default security retention period established by your installation using the RETPD operand on the SETROPTS command.
2. For generic profiles, RACF uses a zero value. This results in the data set being expired. For generic profiles, the default security retention period is *not* checked. Therefore, you must ensure that all generic profiles that protect tape data sets include a retention period. (Make sure to specify the RETPD operand on the **ADDSD** command for generic profiles.)

The security retention period (RETPD) for a tape data set is a number that you specify; *nnnnn* must be one to five digits in the range of 0 - 65533. To indicate a data set that never expires, specify *nnnnn* as 99999.

Figure 3-50 on page 90 shows RVARY and other options for initial setup.

## 3.48 RVARY and other options for initial setup

- RVARY command - background information
  - Importance
  - Authorization required
- Setting the RVARY passwords (RVARYPW)
- Activating JES2 or JES3 RACF Support (JES)
- Establishing national language defaults (LANGUAGE)
- Controlling data set modeling (Model)

Figure 3-50 3.48, "RVARY and other options for initial setup" on page 90

### RVARY command background information

This section provides some background information about the **RVARY** command.

#### ***Importance of the RVARY command***

The **RVARY** command is a very important command for the z/OS system programmer and helpful for maintenance of the RACF database. With the **RVARY** command, you can perform the following functions:

- ▶ Deactivate and reactivate the RACF function.
- ▶ Switch from using a specific primary data set to using its corresponding backup data set, perhaps because of a failure that is related to the primary data set.
- ▶ Deactivate or reactivate primary or backup RACF data sets. (Deactivating a specific primary data set causes all RACF requests for access to that data set to fail. Deactivating a specific backup data set causes RACF to stop duplicating information to that data set.)
- ▶ Deactivate protection for any resources belonging to classes defined in the class descriptor table while RACF is inactive.
- ▶ Select the mode of operation when RACF is enabled for sysplex communication.

#### ***Authorization required***

Unlike the **SETRPTS** command, the **RVARY** command needs no special user attribute for the submitting user ID. However, the operator (at the operator console or security console) must approve the **RVARY** command unless it is an **RVARY LIST** before RACF allows the command to complete.

If the **RVARY** command changes RACF or its database status (**ACTIVE/INACTIVE**), RACF issues an informational message, and the operator is required to enter the password that is defined by **SETRPTS RVARYPW STATUS(status-pw)** to authorize the change.

If the **RVARY** command switches the RACF data sets (**SWITCH**) or changes the RACF operating mode (**DATASHARE/NODATASHARE**), RACF issues an informational message,



and the operator is required to enter the password that is defined by SETROPTS RVARYPW SWITCH(switch-pw).

### Setting the RVARY passwords (RVARYPW)

You use the **SETROPTS** command with the **RVARYPW** parameter to specify the passwords that are necessary for the **RVARY** command to succeed.

RACF allows you to specify separate passwords for switching the databases and for changing RACF status. The following example specifies HAPPY as the switch password and RABBIT as the status password:

```
SETROPTS RVARYPW(SWITCH(HAPPY) STATUS(RABBIT))
```

When RACF is first initialized, the switch password and the status password are both set to YES. Most auditors know to test if the installation has changed these passwords.

**Important:** We strictly recommend changing the RVARY password because of the importance of the command. Otherwise, everyone reading RACF publications can deactivate or influence security in your installation.

### Activating JES2 or JES3 RACF support (JES)

The parameter JES of the **SETROPTS** command has several subcommands that control the job entry subsystem (JES) options. The following subcommands are described in detail in *z/OS Security Server RACF Security's Administrator's Guide, SA23-2289*; and *z/OS Security Server RACF Command Language Reference, SA23-2292*:

<b>BATCHALLRACF</b>	Forcing Batch Users to Identify themselves to RACF
<b>XBMALLRACF</b>	Support for Execution Batch Monitor (XBM) (JES2 Only)
<b>EARLYVERIFY</b>	JES User ID Early Verification
<b>NJEUSER</b>	Understanding Default User IDs
<b>UNDEFINEDUSER</b>	Understanding Default User IDs

These are important settings to impose controls on entities submitting jobs for processing on z/OS. From forcing all incoming users to be validly present on the system to assigning a value to those undefined users that attempt access (instead of “+++++++” you could have them reported as “UNKNOWN” because this may be more useful).

### Establishing national language defaults (LANGUAGE)

With the **LANGUAGE** option of the **SETROPTS** command, you can specify the system-wide defaults for national languages (such as American English or Japanese) that your system uses. You can specify a primary language, a secondary language, or both. The languages that you specify depend on which products, when installed on your system, check for primary and secondary languages (using **RACROUTE REQUEST=EXTRACT**).

To specify the installation default languages, enter:

```
SETROPTS LANGUAGE(PRIMARY(1language1) SECONDARY(1language2))
```

To see the language information only, enter:

```
LISTUSER your-userID LANGUAGE NORACF
```

**Note:** The SETROPTS LANGUAGE operand does not affect the language in which the RACF ISPF panels are displayed. The order in which the RACF ISPF panel libraries are allocated determines the language that is used. If your installation ordered a translated feature of RACF, the z/OS Security Server RACF program directory gives instructions for setting up the ISPF panels.

### **Controlling data set modeling (Model)**

The MODEL operand of the **SETROPTS** command allows you to supplement the information that is normally placed in new data set profiles automatically by ADSP, PROTECT=YES, or ADDSD.

NOMODEL is in effect when a RACF database is first initialized using IRRMIN00.

**Note:** The FROM(profile-name) operand on the **ADDSD** command overrides any specifications from the MODEL(USER) or MODEL(GROUP) operands.

Figure 3-51 on page 93 shows RACF and auditing.

## 3.49 RACF and auditing

- AUDITOR authorization required
- General audit controls for RACF commands
  - AUDIT: Logging RACF commands and DEFINE requests
  - CMDVIOL: Logging RACF command violations
  - SAUDIT: Logging activities of users with the SPECIAL attribute

Figure 3-51 3.49, "RACF and auditing" on page 93

### AUDITOR authorization required

There are several system-wide audit controls using the **SETROPTS** command. General audit controls direct RACF to log (or not to log) certain security-relevant events. To specify the general audit controls, you must have the AUDITOR attribute.

This section describes the auditor control options that refer to security events with RACF commands.

### AUDIT: Logging RACF commands and DEFINE requests

RACF provides means to specify individually for which classes RACF logs all detected accesses to the RACF database through RACF commands and DEFINE requests. You can specify the AUDIT operand on the **SETROPTS** command. Logging becomes effective immediately. The following example specifies that you want RACF to log RACF commands and define requests for users, groups, data sets, and the TERMINAL general-resource classes.

```
SETROPTS AUDIT(USER GROUP DATASET TERMINAL)
```

If you specify AUDIT(\*), logging occurs for all classes.

You deactivate logging for a class by using the NOAUDIT operand. NOAUDIT(\*) is in effect when RACF is using a newly initialized database.

**Note:** If you activate auditing for a class using SETROPTS AUDIT, RACF activates auditing for all classes in the class descriptor table that have the same POSIT value as the class you specify. For details, see *z/OS Security Server RACF Command Language Reference*, SA23-2292.

### CMDVIOL: Logging RACF command violations

A command violation can occur because RACF does not authorize a user to modify a particular profile or to enter a particular operand on a command. If you specify the CMDVIOL operand on the **SETROPTS** command, RACF logs all command violations (except for LISTDSD, LISTGRP, LISTUSER, RLIST, and SEARCH, which are never logged). CMDVIOL is in effect at RACF initialization.

**Tip:** We recommend that you keep CMDVIOL active and cause RACF to log all the command violations that it detects. You can then use the RACF report writer to produce a printed audit trail of command violations. You can determine how many command violations are occurring and which users are causing the violations. A significant number of command violations, especially when RACF is first installed, can indicate the need for more user education. The report can also help you to identify any specific users who are trying persistently to alter profiles without the proper authority.

If you decide to bypass logging of all violations that are detected by RACF commands (except RVARY and SETROPTS, which are always logged) during RACF command processing, you can specify the NOCMDVIOL operand on the **SETROPTS** command as shown in the following example:

```
SETROPTS NOCMDVIOL
```

### **SAUDIT: Logging of activity of users with the SPECIAL attribute**

The SETROPTS option SAUDIT specifies that RACF is to log RACF commands (except LISTDSD, LISTGRP, LISTUSER, RLIST, and SEARCH) issued by users who either had the SPECIAL attribute or who gained authority to issue the command through the group-SPECIAL attribute. SAUDIT is in effect when RACF is using a newly initialized database.

**Tip:** We recommend that you specify SAUDIT because of the powerful commands a SPECIAL user can submit. You can then use the RACF report writer to produce audit reports.

If you decide to bypass this logging (for example, if you are concerned only with how SPECIAL users change profiles and you have AUDIT(\*) in effect), you can use the following command:

```
SETROPTS NOSAUDIT
```

Figure 3-52 on page 95 shows auditor-related options.

## 3.50 Auditor-related options

- AUDITOR authorization required
- General audit controls for resource access:
  - OPERAUDIT: Logging activities of users with the OPERATIONS attribute
  - LOGOPTIONS: Activating auditing for access attempts by class
  - APPLAUDIT: Auditing for APPC/MVS
  - SECLABELAUDIT: Activating auditing for security labels
  - SECLEVELAUDIT: Activating auditing for security levels

Figure 3-52 3.50, "Auditor-related options" on page 95

### **AUDITOR authorization required**

There are further system-wide audit controls using the **SETROPTS** command for which the AUDITOR attribute is needed. This section describes the auditor control options that refer to security events with access to resources.

### **General audit controls for resource access**

The purpose of these controls is to ensure that we capture audit events so that they can be reported on. When an audit event occurs because of a RACF setting, an information record is written to SMF. It is from this collected SMF data that audit reports can be processed or the SMF data can be moved elsewhere such as into DB2 for analysis.

#### ***OPERAUDIT: Logging activities of users with the OPERATIONS attribute***

The SETROPTS option OPERAUDIT specifies that RACF is to audit all accesses to resources granted and all uses of the ADDSD, and RDEFINE commands allowed *only* because the user has the OPERATIONS or group-OPERATIONS attribute. Without the OPERATIONS attribute, the access is denied, because the user is not authorized over the access list. The following example shows how to specify this option:

```
SETROPTS OPERAUDIT
```

NOOPERAUDIT is in effect at RACF initialization.

**Tip:** OPERAUDIT might be useful if you decide to remove the OPERATIONS attribute and give those users access through the normal access list. You can then use the RACF report writer or other auditing tools to produce a report on this event.

#### ***LOGOPTIONS: Activating auditing for access attempts by class***

With the LOGOPTION operand, you can cause RACF to audit attempts of accessing resources in specified classes (whether or not successful). There are different options available. You can specify the DATASET class and any active classes in the class descriptor table. The resources need not have profiles created in order for the auditing to occur. The

following command specifies that auditing is to be done for all attempts to access the TERMINAL class:

```
SETROPTS LOGOPTIONS(ALWAYS(TERMINAL))
```

In this case, auditing is done every time that a user logs on at any terminal on the system, regardless of whether that terminal is protected by a profile and regardless of whether that profile specifies auditing. You can specify that auditing be done for the following conditions:

- ALWAYS** All attempts to access resources protected by the class are audited.
- NEVER** No attempts to access resources protected by the class are audited. (All auditing is suppressed.)
- SUCCESSSES** All successful attempts to access resources protected by the class are audited.
- FAILURES** All failed attempts to access resources protected by the class are audited.
- DEFAULT** Auditing is controlled by the profile protecting the resource, if a profile exists. You can specify DEFAULT for all classes by specifying an asterisk (\*) with DEFAULT.

**Note:**

- ▶ The SUCCESSSES and FAILURES operands result in auditing *in addition to* any auditing that is specified in profiles in the class. In contrast, the ALWAYS and NEVER operands *override* any auditing specified in profiles in the class.
- ▶ When RACF grants access to a resource because of an entry in the *global access checking table*, RACF does not log the event even if you request logging.

LOGOPTIONS(DEFAULT(\*)) is in effect at RACF initialization.

### **APPLAUDIT: Auditing for APPC/MVS**

Specifying the APPLAUDIT parameter on the SETROPTS command, you can request auditing of APPC transactions. NOAPPLAUDIT is in effect at RACF initialization. If this is in effect and if the class APPL has the audit setting on and the application does not support persistent verification, it may cause a large amount of data to be written to SMF. Persistent verification (PV) is where the security environment for a user is created when the user's *first* transaction request enters the system. The security environment *persists* over multiple transactions before being deleted. See *z/OS Security Server RACF Auditor's Guide*, SA23-2290, for more information.

### **SECLABELAUDIT: Activating auditing for security labels**

The SECLABELAUDIT option of the SETROPTS command specifies that the SECLABEL profile's auditing options are to be used in addition to the auditing options specified for the user or resource. NOSECLABELAUDIT is in effect when RACF is using a newly initialized database. For more information, refer to *z/OS Security Server RACF Auditor's Guide*, SA23-2290.

### **SECLEVELAUDIT: Activating auditing for security levels**

The SECLEVELAUDIT (security-level) parameter of the SETROPTS command activates auditing of access attempts to all RACF protected resources based on the specified installation-defined security level. This is used to log accesses to resources that are protected by a security level. RACF audits all access attempts for the specified security level and higher. You can specify only a security level name defined by your installation as a SECLEVEL profile in the SECDATA class. (For information about defining security levels, see the description of the

RDEFINE and RALTER commands in *z/OS Security Server RACF Command Language Reference*, SA23-2292.)

The NOSECLEVELAUDIT operand deactivates auditing of access attempts to RACF protected resources based on a security level. NOSECLEVELAUDIT is in effect when RACF is using a newly initialized database.

Figure 3-53 shows the SETROPTS LIST command.

## 3.51 SETROPTS: Display options (LIST)

▪ SETROPTS LIST example:

```
ATTRIBUTES = INITSTATS WHEN(PROGRAM -- BASIC)
STATISTICS = DATASET DASDVOL GDASDVOL NODES NODMBR TAPEVOL
ACTIVE CLASSES = DATASET USER GROUP ACCTNUM ACICSPCT AIMS APPCLU APPL
...
GENERIC PROFILE CLASSES = DATASET @GILL ACCTNUM ACICSPCT AIMS ALCSAUTH
...
GENLIST CLASSES = LOGSTRM
GLOBAL CHECKING CLASSES = DATASET ACCTNUM ACICSPCT AIMS APPCLU APPCPORT
                           APPCSERV APPCSI APPCTP APPL CCICSCMD CIMS CONSOLE
...
GLOBAL=YES RACLIST ONLY = NONE
AUTOMATIC DATASET PROTECTION IS NOT IN EFFECT
ENHANCED GENERIC NAMING IS IN EFFECT
REAL DATA SET NAMES OPTION IS INACTIVE
JES-BATCHALLRACF OPTION IS INACTIVE
JES-XBMALLRACF OPTION IS INACTIVE
JES-EARLYVERIFY OPTION IS INACTIVE
PROTECT-ALL OPTION IS NOT IN EFFECT
TAPE DATA SET PROTECTION IS INACTIVE
SECURITY RETENTION PERIOD IN EFFECT IS      0 DAYS.
ERASE-ON-SCRATCH IS INACTIVE
SINGLE LEVEL NAMES NOT ALLOWED
LIST OF GROUPS ACCESS CHECKING IS ACTIVE.
INACTIVE USERIDS ARE NOT BEING AUTOMATICALLY REVOKED.
NO DATA SET MODELLING BEING DONE.
PASSWORD PROCESSING OPTIONS:
...
```

Figure 3-53 3.51, "SETROPTS: Display options (LIST)" on page 97

### SETROPTS LIST command

This command specifies that the current RACF options are displayed. If you specify operands in addition to LIST on the SETROPTS command, RACF processes the other operands before it displays the current set of options.

If RACF is enabled for sysplex communication and the system is in read-only mode, users on that system can issue the SETROPTS LIST command. All other operands are ignored.

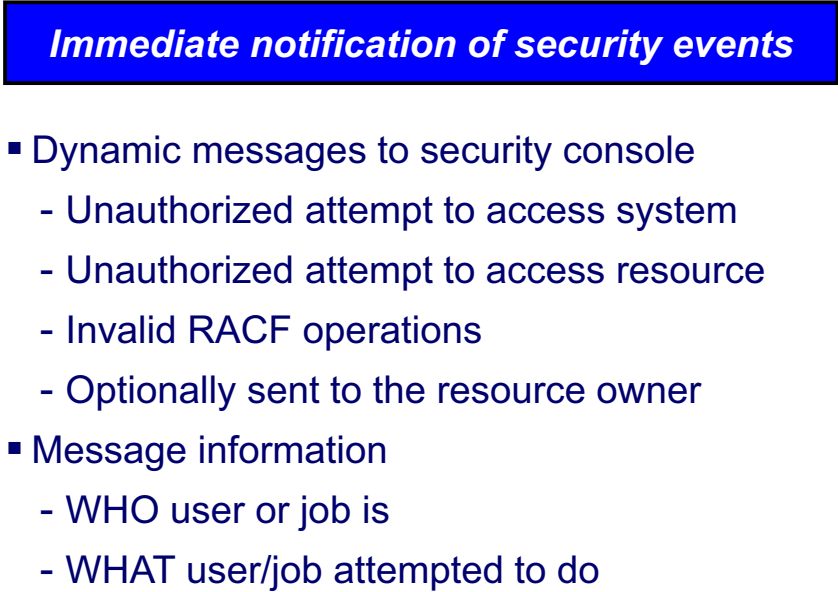
You must have the RACF SPECIAL, AUDITOR, group-SPECIAL, or group-AUDITOR attribute to enter the LIST operand.

If you have the SPECIAL or group-SPECIAL attribute, and not the AUDITOR or group-AUDITOR, RACF displays all operands except the auditing-related operands.

Figure 3-53 on page 97 shows portions of the output from the following **SETROPTS** command:  
SETROPTS LIST

Figure 3-54 shows RACF monitoring.

## 3.52 RACF monitoring



**Immediate notification of security events**

- Dynamic messages to security console
  - Unauthorized attempt to access system
  - Unauthorized attempt to access resource
  - Invalid RACF operations
  - Optionally sent to the resource owner
- Message information
  - WHO user or job is
  - WHAT user/job attempted to do

Figure 3-54 RACF monitoring

### RACF monitoring

Security Server RACF routes system operator messages to a system console or a security console. The message suffix indicates if an action is required:

1. A: An action is required, the operator must perform a specific action.
2. D: A decision is required, the operator must choose an alternative.
3. E: An eventual action is needed to be done.
4. I: This is an informational message and no action is required.
5. W: It is a wait, and processing for this task stops until action is determined and performed.

The example in Figure 3-55 on page 99 shows a sample RACF message on the system console. Because the number of violations for a large network can be high due to misspelled passwords, transaction codes, and commands, you can specify a threshold for notification. The master terminal is not notified until the specified number of violations occurs without a valid input from a given terminal. You specify one to three invalid entries as the violation limit, eliminating or reducing the number of notifications that are caused merely by operator error, while still providing evidence of real attempts to avoid security safeguards.



```

ICH408I USER(JAMES ) GROUP(MFG ) NAME(BROWN JAMES )
SUPERUSER.FILESYS.MOUNT CL(UNIXPRIV)
DEFINE - RESOURCE ALREADY DEFINED

```

Figure 3-55 RACF message on system console

Figure 3-56 shows a RACF monitoring example.

### 3.53 RACF monitoring, continued

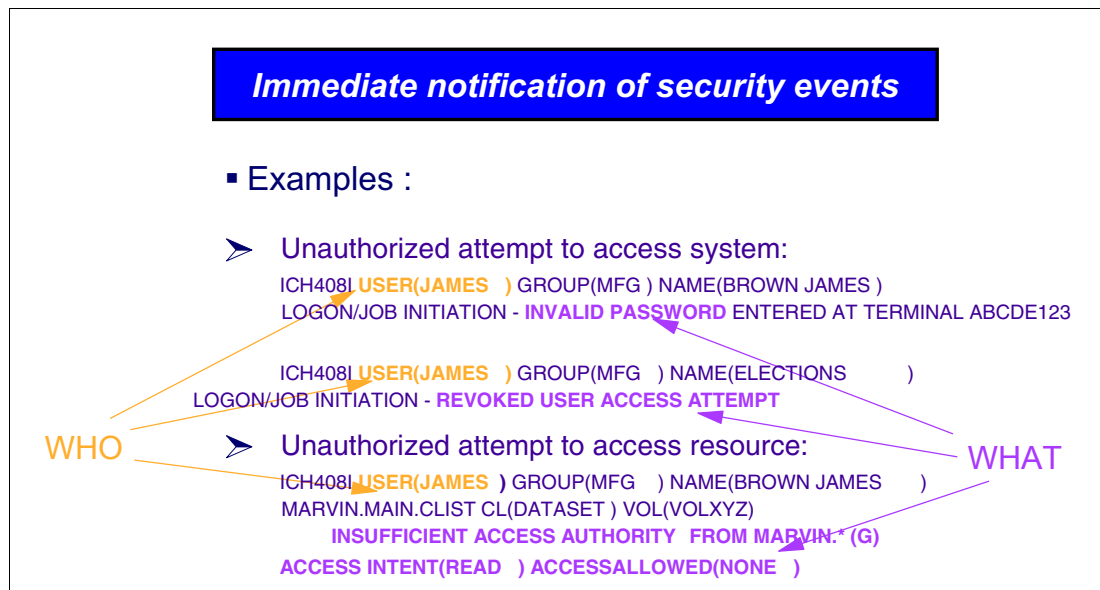


Figure 3-56 RACF monitoring example

#### Example of RACF immediate notification: Example 1

The explanation of the RACF message ICH408I is as follows:

```

ICH408I USER(userid) GROUP(group-name) NAME(user-name)

```

This message is issued when RACF detects an unauthorized request (a violation) made by a user or job. The user and group indicated in the first line of the ICH408I message are the execution user ID and group ID under which the job was to run.

The message in Figure 3-56 shows who the user ID is. If the user ID is shown as the form of “\*\*nnxxxx”, such as “\*\*01XUSR”, the user ID identifies an identity context reference, not a RACF user ID.

For the “what”, we look at the explanatory text within the message, which shows these causes:

1. User ID JAMES entered an invalid password at a terminal
2. User ID JAMES exceeded the number of times an invalid password entry is permitted and is revoked
3. User ID JAMES tried to access a data set to read it but was not allowed to do so

Detailed information about the violation is available in the SMF type 80 record that RACF produces at the same time as this message.

Figure 3-57 shows a RACF immediate notification example.

### 3.54 RACF monitoring, continued

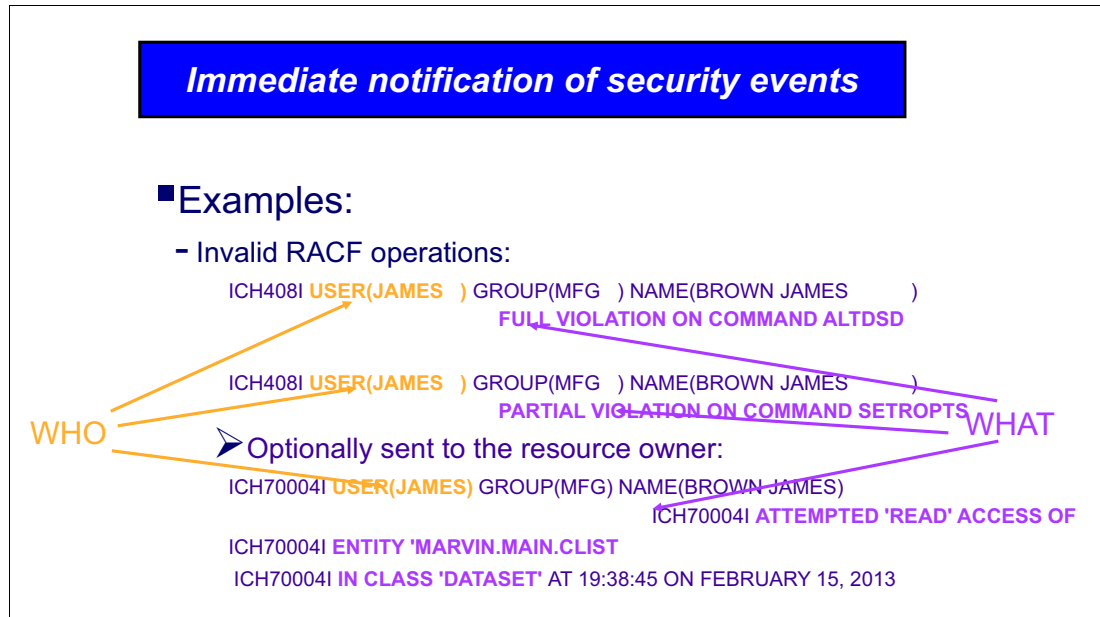


Figure 3-57 RACF immediate notification example

#### Example of RACF immediate notification: Example 2

The RACF message ICH70004I is as follows:

```
ICH70004I USER(accessor) GROUP(group-name)
NAME(user-name) ATTEMPTED 'access-type' ACCESS
OF ENTITY 'resource-name' IN CLASS 'class-name' AT
hh:mm:ss ON month day, year.
```

This message alerts a RACF user that an access violation has occurred against the indicated resource. This message is routed to the user specified in the NOTIFY field of the resource profile that denied the access.

Figure 3-58 on page 101 shows RACF auditing tools.

## 3.55 RACF auditing tools

### *Delayed investigations about security events*

- Auditing Tools:
  - SMF Data Unload Utility (IRRADU00)
  - RACF Report Writer (RACFRW)
  - Data Security Monitor (DSMON)
  - RACF Data Unload Utility (IRRDBU00)

Figure 3-58 RACF auditing tools

### **RACF auditing tools**

RACF auditing is basically verifying that the principals set forth by the installations security policy are not compromised. The challenge with auditing is to reduce the amount of information to something that can be easily analyzed.

Two types of auditing data exist:

- ▶ Security data content from the RACF database, which is a static image or a snapshot of the system parameters at any one time.
- ▶ Security events data statistical information, such as the date, time, and the number of times a specific resource was accessed by any one user.

RACF writes security log records when it detects:

- ▶ Unauthorized attempts to enter the system
- ▶ Authorized or unauthorized attempts to enter RACF commands
- ▶ RACF status changes
- ▶ Warning mode resource access attempts
- ▶ Failsoft operator access decisions
- ▶ Optional authorized or unauthorized attempts to access RACF protected resources

These security log records are commonly called SMF records; some types include:

- |                |   |
|----------------|---|
| <b>Type 80</b> | Produced during RACF processing.  |
| <b>Type 81</b> | Produced at the completion of RACF initialization and from the <b>SETROPTS</b> command.   |
| <b>Type 83</b> | Produced during RACF and z/OS component processing. The type 83 record is also generated under the following circumstances: SETROPTS MLACTIVE is in effect and a RACF command ( <b>ADDSD</b> , <b>ALTDSD</b> , <b>DELDSD</b> ) has been issued that changed the security label of a data set profile. Also, from security events detected by non RACF components. |

You can list the contents of these records to help you to detect possible security exposures or threats and verify the security of the system.

Each of the following programs can help you accomplish your goals, depending on your specific needs:

- ▶ SMF data unload utility
- ▶ RACF report writer
- ▶ Data security monitor (DSMON)
- ▶ RACF data unload utility

Figure 3-59 shows the SMF unload utility.

## 3.56 RACF auditing - IRRADU00

- **SMF Unload Utility (IRRADU00 program):**
  - Enables creation of sequential file from security relevant audit events
  - Allows processing of complex inquiries on SMF records in different ways
  - Allows creation of installation-tailored reports

*Figure 3-59 SMF unload utility*

### **SMF data unload utility (IRRADU00 program)**

The system management facility (SMF) data unload utility processes SMF records and permits more complex auditing than the RACF report writer. Output from the SMF data unload utility can be:

- ▶ Viewed directly
- ▶ Used as input for installation-written programs
- ▶ Manipulated by sort or merge utilities
- ▶ Uploaded to a database manager, such as DB2

You can process complex inquiries and generate custom-tailored reports from the output of the SMF data unload utility. These reports can be useful in identifying suspicious patterns of access by authorized users that another program might miss. Because data is more often misused by authorized users than stolen by unauthorized users, reports such as this are essential to security.

A sample from this utility is shown on Figure 3-60 on page 103.

▪ SMF Unload Utility output example:

```

JOBINIT SUCCESS 13:31:44 1996-10-08 9672 ADMIN
JOBINIT SUCCESS 13:31:46 1996-10-08 9672 ADMIN
JOBINIT SUCCESS 13:31:47 1996-10-08 9672 ADMIN
JOBINIT TERM 13:31:48 1996-10-08 9672 ADMIN
JOBINIT TERM 13:31:48 1996-10-08 9672 ADMIN
JOBINIT TERM 13:31:48 1996-10-08 9672 ADMIN
SETROPTS INSAUTH 13:39:12 1996-10-08 9672 YES NO NO JAMES MFG NO NO NO NO
ALTUSER SUCCESS 13:40:12 1996-10-08 9672 NO NO NO JAMES MFG NO YES NO NO
JOBINIT SUCCESS 13:40:46 1996-10-08 9672 JAMES MFG
SETROPTS SUCCESS 13:41:46 1996-10-08 9672 NO NO NO JAMES MFG NO NO NO YES
SETROPTS SUCCESS 13:42:17 1996-10-08 9672 NO NO NO JAMES MFG NO YES NO YES
DIRSRCH SUCCESS 13:43:11 1996-10-08 9672 NO NO NO ARTHUR FINC YES NO NO NO
DIRSRCH SUCCESS 13:43:11 1996-10-08 9672 NO NO NO ARTHUR FINC YES NO NO NO
DIRSRCH NOTAUTH 13:43:11 1996-10-08 9672 YES NO NO ARTHUR FINC YES NO NO NO
CHKFOWN OWNER 13:44:28 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO
CHMOD SUCCESS 13:44:28 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO
CHOWN SUCCESS 13:44:28 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO
FACCESS SUCCESS 13:44:28 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO
JOBINIT SUCCESS 13:44:30 1996-10-08 9672 REBECCA
SETEUD SUCCESS 13:44:31 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO
DIRSRCH SUCCESS 13:44:31 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO
DIRSRCH SUCCESS 13:44:31 1996-10-08 9672 NO NO NO REBECCA SYS1 NO NO NO NO

```

SMF Unload Utility

Figure 3-60 SMF unload utility example

### How to run the SMF data unload utility (IRRADU00)

To perform this function, the SMF dump utility IFASMFDP is invoked in a batch job. The RACF SMF data unload utility is specified in the USER2 and USER3 exits.

Following is a RACF SMF data unload utility sample JCL:

```

//KHEWITT1 JOB (ITS0),'SMF FLAT',MSGCLASS=X
//SMFDUMP EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=*
//ADUPRINT DD SYSOUT=*
//OUTDD DD DISP=(NEW,CATLG),DSN=KHEWITT.RACF.IRRADU00,
// UNIT=SYSDA,SPACE=(CYL,(10,5),RLSE),
// LRECL=12288,RECFM=VB,BLKSIZE=12292,
//SMFDATA DD DISP=SHR,DSN=SYS1.SC42.MAN2
//SMFOUT DD DUMMY
//SYSIN DD *
        INDD(SMFDATA,OPTIONS(DUMP))
        OUTDD(OUTDD,TYPE(000:255))
        ABEND(NORETRY)
        USER2(IRRADU00)
        USER3(IRRADU86)
/*

```

If the OUTDD format is not suitable, see the *z/OS Security Server RACF Auditor's Guide*, SA23-2290, about how to set up XMLFORM DD (output data in XML format) or XMLOUT (output data in compressed XML format), and other information about this data unload utility.

To determine the input file for the SMFDATA DD, which typically is the active SMF data set, display the active SMF data set using the **D SMF** command from the system console as follows:

```
IEE974I 10.12.27 SMF DATA SETS 796
      NAME                VOLSER SIZE(BLKS) %FULL  STATUS
P-SYS1.SC42.MAN1        MVS004   1200    0  ALTERNATE
S-SYS1.SC42.MAN2        MVS004   1200   86  ACTIVE
S-SYS1.SC42.MAN3        MVS004   1200    0  ALTERNATE
```

MAN2 is the active SMF data set. The output file in this example is KEWITT.RACF.IRRADU00.

**Note:** Due to restrictions of the SMF dump utilities, IRRADU00 and IRRADU86 must reside in an APF-authorized library.

Figure 3-61 shows the RACF report writer.

### 3.57 RACF auditing: RACF Report Writer

- RACF Report Writer (RACFRW):
  - Allows report generation from SMF records

Figure 3-61 RACF report writer

#### RACF report writer

The RACF report writer lists the contents of SMF records in a format that is easy to read. It also uses the same SMF data to generate the following specialized reports:

- ▶ Reports that describe attempts to access a particular RACF protected resource in terms of user identity, number, and type of successful accesses, and number and type of attempted security violations.
- ▶ Reports that describe user and group activity.
- ▶ Reports that summarize system use and resource use.

**Note:** The report writer is *no longer* the recommended utility for processing RACF audit records. The RACF SMF data unload utility is the preferred reporting utility. The report writer does not support all of the audit records introduced after RACF 1.9.2.

Figure 3-62 on page 105 shows a RACF report writer example.

▪ RACF Report Writer report example:

```
99.049 00:24:29                                RACF REPORT
                                                ACCESSES TO MYMUSIC PROGRAM
COMMAND GROUP ENTERED -
RACFRW          LINECNT(60)                    FORMAT          GENSUM
SELECT PROCESS
EVENT ACCESS CLASS(PROGRAM) NAME(MYMUSIC )
LIST TITLE('LIST ACCESSES TO MYMUSIC PROGRAM')
SUMMARY RESOURCE BY(USER)  TITLE('SUMMARY BY USER')
END
```

.../...

**RACF Report Writer**

Figure 3-62 RACF report writer example

### How to run RACF report writer

A RACF report writer sample JCL is shown in Figure 3-63.

```
//RACFRW2 EXEC PGM=IKJEFT01
//SORTWKxx DD your sort work files
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//RSMFIN DD DISP=(SHR,PASS,DELETE),DSN=*.SMFDUMP.QSAMOUT
//SYSTSIN DD *,DLM=XX
RACFRW TITLE('RACF REPORTS') GENSUM
SELECT VIOLATIONS
LIST TITLE('ACCESS VIOLATIONS SUMMARY REPORT')
SUMMARY RESOURCE BY(USER)
END
XX
/*
```

Figure 3-63 Sample JCL to run RACFRW processing

The RACF report writer can also be run from the TSO command line using the **RACFRW** command. In the example shown in Figure 3-63, the SMF data is to be found in the RSMFIN DD statement.

Figure 3-64 on page 106 shows the RACF data security monitor (DSMON).

## 3.58 RACF auditing: DSMON

- DSMON = System Integrity Validation Tool
  - Shows current status of data security & system integrity
  - Generates optional set of reports:
    - System report
    - Selected data set report
    - Program properties table report (z/OS only)
    - Selected user attribute reports
    - RACF EXITS report
    - Started procedure table (z/OS only)
    - Class Descriptor Table
    - Global Access Checking table
    - Group tree report

*Figure 3-64 RACF data security monitor*

### **RACF data security monitor**

The RACF data security monitor (DSMON) enables you to verify the basic system integrity and data security controls.

RACF auditors can use the DSMON reports to evaluate the level of security at the installation and to compare the actual level of security at an installation with the planned level of security. If the installation has not defined ICHDSM00 (DSMON) as a controlled program, you must have the AUDITOR attribute to run DSMON.

### **DSMON reports**

DSMON produces the reports that are shown in Figure 3-65 on page 107.



- System Report
- Group Tree Report
- Program Properties Table Report
- RACF Authorized Caller Table Report
- RACF Class Descriptor Table Report
- RACF Exits Report
- RACF Global Access Checking Table Report
- RACF Started Procedures Table Report
- RACF User Attribute Report
- RACF User Attribute Summary Report
- Selected Data Sets Reports

Figure 3-65 DSMON reports

### The system report

The system report contains information such as the identification and model of the processor complex, and the name, version, and release of the operating system. This report also specifies the RACF version and release number and whether RACF is active. If RACF is inactive, DSMON prints a message that tells you whether RACF was not activated at IPL or was deactivated by the **RVARY** command. It should be noted that some of these reports are available when RACF is used on z/VM. The reports that are specific to z/VM are marked as such.

```

RACF DATA SECURITY MONITOR
S Y S T E M   R E P O R T
-----
CPU-ID                01B8D7
CPU MODEL             2827
OPERATING SYSTEM/LEVEL z/OS 2.1.0
SYSTEM RESIDENCE VOLUME Z21RC1
SMF-ID                SC74
RACF (FMID HRF7790) IS ACTIVE

```

Figure 3-66 System report from DSMON

The job control to produce these reports is straightforward. You can specify DSMON control statements to produce the reports that you want and control the number of lines per page for each report. The output from DSMON consists of a message data set and an output data set for the reports. This is shown in the JCL in Figure 3-67 on page 108, which produced a system report. Refer to *z/OS Security Server RACF Auditor's Guide*, SA23-2290, for the full description.

```

//REPORT EXEC PGM=ICHDSMOO
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD SYSOUT=*
//SYSIN DD *
          FUNCTION SYSTEM
/*

```

Figure 3-67 Sample JCL for DSMON

### The group tree report

This report lists, for each requested group, all of its subgroups, all of the subgroups of the subgroups, and so on, as well as the owner of each group that is listed in the report, if the owner is not the superior group. You can use the group tree report to examine the overall RACF group structure for your system. You can also determine the scope of the group for group-related user attributes (group SPECIAL, group OPERATIONS, and group AUDITOR).

So if we use a FUNCTION RACGRP control statement we obtain the report as shown in Figure 3-68.

RACF DATA SECURITY MONITOR			R A C F   G R O U P   T R E E		
LEVEL	GROUP	(OWNER)			
1	SYS1	(IBMUSER )			
2	AOPADMIN	(ROGERS )			
2	AOPOPER	(ROGERS )			
2	ARADMIN	(ARUN )			

Figure 3-68 Small portion of the FUNCTION RACGRP report

### The program properties table report

This report lists all of the programs in the z/OS program properties table (PPT). The report also indicates, for each program, whether the program is authorized to bypass password protection and whether it runs in a system key.

You can use the program properties table report to verify that only those programs that the installation has authorized to bypass password protection are, in fact able to do so. Such programs are normally communication and database control programs and other system control programs. These are what are called *trusted resource managers*.

You can also verify that only those programs that the installation has authorized are able to run in a system key. So if we use a FUNCTION SYSPPT control statement, we obtain the report as shown in Figure 3-69 on page 109.

RACF DATA SECURITY MONITOR		
PROGRAM	BYPASS PASSWORD	PROGRAM PROPERTIES
NAME	PROTECTION	SYSTEM KEY
IEDQTCAM	NO	YES
ISTINM01	YES	YES
IKTCAS00	NO	YES
AHLGTF	NO	YES
HHLGTF	NO	YES
IHLGTF	NO	YES
IEFIIC	NO	YES
IEEMB860	YES	YES

Figure 3-69 Partial report of FUNCTION SYSPT

### The RACF authorized caller table report

This report lists the names of all of the programs in the RACF authorized-caller table. The programs in this table are authorized to issue REQUEST=VERIFY (which performs user verification) or REQUEST=LIST (which loads profiles into main storage).

You can use this report to verify that only those programs that are supposed to be authorized to modify an ACEE (accessor environment element) are able to issue a REQUEST=VERIFY. This verification is a particularly important security requirement because the ACEE contains a description of the current user. This description includes the user ID, the current connect group, the user attributes, and the group authorities. A program that is authorized to issue REQUEST=VERIFY could alter the ACEE to simulate any user.

You can also use this report to verify that only those programs that are supposed to be authorized to access profiles are able to issue REQUEST=LIST. Because profiles contain complete descriptions of the characteristics that are associated with RACF defined entities, you must carefully control access to them.

**Note:** IBM does not recommend using the RACF authorized caller table.

### The RACF class descriptor table report

This report lists, for each general resource class the class name, the default UACC, whether the class is active, whether auditing is being done, whether statistics are being kept, and whether OPERATIONS attribute users have access.

You can use the class descriptor table report to determine which classes (in addition to DATASET) are defined to RACF and active and, therefore, can contain resources that RACF protects. So if we use a FUNCTION RACCDT control statement, we obtain the report as shown in Figure 3-70 on page 110.

RACF DATA SECURITY MONITOR				
CLASS		R A C F	C L A S S	D E S C R I P T O R
NAME	STATUS	AUDITING	STATISTICS	DEFAULT UACC
PSFMPL	INACTIVE	NO	NO	NONE
PTKTDATA	ACTIVE	NO	NO	NONE
PTKTVAL	ACTIVE	NO	NO	ACEE
QCICSPSB	ACTIVE	NO	NO	NONE
QIMS	INACTIVE	NO	NO	NONE
RACFEVNT	ACTIVE	NO	NO	NONE
RACFHC	INACTIVE	NO	NO	NONE

Figure 3-70 Partial list from FUNCTION RACCDT

It should be noted in this report that none of the active classes are being audited. Normally it would be expected that auditing is active for these classes.

### The RACF exits report

This report lists the names of all of the installation-defined RACF exit routines and specifies the size of each exit routine module.

You can use the RACF exits report to verify that the only active exit routines are those that your installation has defined. The existence of any other exit routines might indicate a system security exposure because RACF exit routines can be used to bypass RACF security checking. Similarly, if the length of an exit routine module differs from the length of the module when it was defined by your installation, the module might have unauthorized modifications.

So if we use a FUNCTION RACEXT control statement, we obtain the report as shown in Figure 3-71. This shows that this installation has implemented no RACF exits.

RACF DATA SECURITY MONITOR		R A C F	E X I T S
EXIT MODULE	MODULE		
NAME	LENGTH		
-----			
NO RACF EXITS ARE ACTIVE			

Figure 3-71 List from FUNCTION RACEXT

### The RACF global access checking table report

This report lists, for each resource class in the global access table, all of the entry names and their associated resource access authorities.

Because global access checking allows anyone to access the resource at the associated access authority, you should verify that each entry has an appropriate level of access authority.

For started procedures, RACF generates two reports about the started procedures table.

If the STARTED class is active, the report uses the STARTED class profiles and contains the TRACE attribute. The trace uses module ICHDSM00.

If the STARTED class is not active, the trace uses the installation replaceable load module, ICHRIN03.

The reports list the procedure name, the user ID, and group name to be associated with the procedure, and whether the procedure is privileged or trusted.

You can use the report to determine which started procedures are defined to RACF, and which have the privileged attribute. If a started procedure is privileged or trusted, it bypasses all REQUEST=AUTH processing (unless the CSA or PRIVATE operand was specified on REQUEST=AUTH), including checks for security classification of users and data.

So if we use a FUNCTION RACGAC control statement, we obtain the report as shown in Figure 3-72. This shows for these entries the installation has no active classes for global access checking.

RACF DATA SECURITY MONITOR			
GLOBAL	ACCESS	TABLE	REPORT
CLASS	ACCESS	ENTRY	
NAME	LEVEL	NAME	
-----			
DATASET		-- NO ENTRIES --	
TAPEVOL		-- NO ENTRIES --	
TERMINAL		-- NO ENTRIES --	
AFACILITY		-- NO ENTRIES --	
UNIXPRIV		-- GLOBAL INACTIVE --	
VMMDISK		-- NO ENTRIES --	

Figure 3-72 Partial list of FUNCTION RACGAC

### Started procedures report

The started procedures table report lists each entry in the started procedures table. Each entry contains the procedure name, user identification, the group name associated with the procedure, the privileged status, and the trusted status. If the STARTED class is active, the report that gets generated also shows the job name associated with the procedure and the TRACE attribute.

So if we use a FUNCTION RACSPT control statement, we obtain the report as shown in Figure 3-73 on page 112.

```

RACF DATA SECURITY MONITOR
                R A C F      S T A R T E D      P R O C E D U R E S
FROM PROFILES IN THE STARTED CLASS:
-----
PROFILE          ASSOCIATED  ASSOCIATED
NAME            USER        GROUP        PRIVILEGED  TRUSTED  TRACE
-----
DIRECT.PLEX76   -STDATA NOT SPECIFIED, ICHRIN03 WILL BE USED-
IZUANG1.IZUANG7 IZUS75   IZUADM75   NO          NO        NO
IZUSVR1.IZUSRV7 IZUS75   IZUADM75   NO          NO        NO
IZUSVR1.IZUSVR7 IZUS75   IZUADM75   NO          NO        NO
MVS*           =MEMBER
ANTAS000.* (G)  ANTAS000 STCGROUP   NO          YES       YES
ANTMAIN.* (G)  ANTMAIN  STCGROUP   NO          YES       YES
AOP.* (G)      AOPSTC   AOOPER     NO          NO        NO
AOPSTART.* (G) VAINI    SYS1       YES         YES       YES

```

Figure 3-73 Partial list of FUNCTION RACSPT

### Selected user attribute report

The selected user attribute report:

- ▶ Lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, or REVOKE attributes
- ▶ Specifies whether they possess these attributes on a system-wide (user) or group level
- ▶ Indicates whether they have any user ID associations

You can use this report to verify that only those users who need to be authorized to perform certain functions have been assigned the corresponding attribute.

So if we use a FUNCTION RACUSR control statement, we obtain the report as shown in Figure 3-74.

```

RACF DATA SECURITY MONITOR
S E L E C T E D      U S E R      A T T R I B U T E
-----
USERID              ----- ATTRIBUTE TYPE -----
                   SPECIAL      OPERATIONS      AUDITOR
-----
ALEXL               SYSTEM          SYSTEM
ANDYK               SYSTEM          SYSTEM
ANHORN              SYSTEM          SYSTEM
ARAMACH             SYSTEM          SYSTEM
ARUN                SYSTEM          SYSTEM
BECKER              SYSTEM          SYSTEM
BIRD                SYSTEM          SYSTEM
BOBMCC              SYSTEM          SYSTEM
BTANGUY             SYSTEM          SYSTEM
BURGER              SYSTEM          SYSTEM
BUZZO               SYSTEM
CAMARGO             SYSTEM          SYSTEM          SYSTEM

```

Figure 3-74 Partial list of FUNCTION RACUSR; some empty report columns have been omitted

## Selected user attribute summary report

The selected user attribute summary report shows the number of installation-defined users and totals for users with the SPECIAL, OPERATIONS, AUDITOR, and REVOKE attributes, at both the system and group level. You can use this report to verify that the number of users with each of these attributes, on either a system or group level, is the number that your installation wants. In particular, you should make sure that you have assigned the SPECIAL attribute (on a system level) to at least one user and the AUDITOR attribute (on a system level) to at least one user.

So if we use a FUNCTION RACUSR control statement the following report is also produced as shown in Figure 3-75.

RACF DATA SECURITY MONITOR			
S E L E C T E D		U S E R A T T R I B U T E	
TOTAL DEFINED USERS:		293	
TOTAL SELECTED ATTRIBUTE USERS:			
ATTRIBUTE BASIS	SPECIAL	OPERATIONS	AUDITOR
SYSTEM	167	163	6
GROUP	0	0	1
OPERATIONS	AUDITOR	REVOKE	
163	6	2	
0	1	0	

Figure 3-75 Summary from FUNCTION RACUSR, some report columns have been moved to fit

## Selected data sets report

This report lists the names of selected system data sets and, for each data set, specifies the criterion for selection, the serial number of the volume on which it resides, whether the data set is RACF indicated or RACF protected, and the universal access authority (UACC). If a data set meets more than one selection criterion, there is a separate entry in the report for each criterion. The selected data sets include system data sets, the z/OS master catalog, user catalogs, the RACF primary and backup data sets, and user-specified data sets.

You can use the selected data sets report to determine which of these data sets are protected by RACF and, which are not. You can also check whether the UACC associated with each of the data sets is compatible with your installation's resource access control requirements.

The available *Selected Data Sets Reports* reports are listed here:

<b>SYSLNK</b>	All LNKSTxx data set members of the SYS1.PARMLIB library.
<b>SYSAPF</b>	Current Link List Data Set Report showing authorized program facility (APF) libraries.
<b>SYSCAT</b>	Selected Data Sets listing the Report Master catalog and all user catalogs.
<b>RACDST</b>	Report showing primary and backup RACF databases.
<b>SYSSDS</b>	Selected system data sets.
<b>USRDSN</b>	Selected user data sets report (used with USEROPTS control statement).

Figure 3-76 on page 114 shows the RACF database unload utility.

## 3.59 RACF auditing: IRRDBU00

- RACF Database Unload Utility (IRRDBU00 program):
  - Enables creation of sequential file from the RACF database
  - Allows processing of complex inquiries on RACF records in different ways
  - Allows creation of installation-tailored reports
  - Can also be uploaded to a database manager

*Figure 3-76 RACF database unload utility*

### **RACF database unload utility**

The RACF database unload utility (IRRDBU00 program) is used to unload data from the RACF database (except password fields) into a sequential file.

The output file from the database unload utility can be:

- ▶ Viewed directly
- ▶ Used as input to your own programs
- ▶ Manipulated with sort and merge utilities
- ▶ Used as input to a database management system

Installations can produce reports that are tailored to their requirements.

You can use the DB2 load utility or its equivalent to process the records that are produced by the database unload utility. The definition and control statements for loading the database unload utility output into a DB2 table space is shipped with z/OS in the SYS1.SAMPLIB(RACDBUTB) member.

When the utility runs, it writes a log of its activities to SYSPRINT, as shown in Figure 3-77 on page 115.



```

Specified option: NOLOCKINPUT
Option in effect: NOLOCKINPUT
SYS1.RACFBK associated with DD INDD1 has been successfully opened.
The blocksize was taken from DD INDD1 and the data set was closed.
Processing 1 RACF data set(s).
SYS1.RACFBK associated with DD INDD1 has been successfully opened.
INDD1 is a backup data set. All input data sets must be backup data sets.
Processing group profiles.
72 group profile(s) have been unloaded.
Processing user profiles.
293 user profile(s) have been unloaded.
Processing dataset profiles.
205 dataset profile(s) have been unloaded.
Processing general profiles.
2 general ACCTNUM profile(s) have been unloaded.

```

Figure 3-77 Partial list from SYSPRINT from running IRRDBU00

### How to run IRRDBU00

A RACF data unload utility sample JCL is shown in Figure 3-78. For the *least* impact to system performance, use a copy of your RACF database as input and specify the **NOLOCKINPUT** parameter. The output file name in this example is BOBMCC.RACFDBU.SC75.

```

//UNLOAD EXEC PGM=IRRDBU00,PARM='NOLOCKINPUT'
//INDD1 DD DISP=SHR,DSN=SYS1.RACFBK
//OUTDD DD DSN=BOBMCC.RACFDBU.SC75,
//      DISP=(NEW,CATLG),
//      UNIT=SYSDA,
//      SPACE=(TRK,(20,10),RLSE),
//      DCB=(RECFM=VB,LRECL=4096)
//SYSPRINT DD SYSOUT=*

```

Figure 3-78 Sample JCL to unload a RACF database into a sequential file

The data written to the sequential file can be viewed directly as shown in Figure 3-79.

JAMES	MFG																			
JAMES	MFG	2013-07-24	ADMUSERS						NONE	00000	NO	NO								
JAMES		2013-07-24	ADMUSERS	NO	NO	NO	YES	NO	090	BROWN	JAMES									

Figure 3-79 Small extract from database unload sequential data set

Figure 3-80 on page 116 shows CDT actions.

## 3.60 RACF and Dynamic CDT

- Maintain installation defined classes
- Report on installation defined classes

Figure 3-80 CDT actions

### Maintain installation defined classes

The class descriptor table contains the names and attributes of the resource classes that can be used on your RACF system. The dynamic CDT contains optional CDT entries that are created when you define profiles in the CDT general resource class. Therefore, the names of these profiles in the CDT class become the names of the new classes in the dynamic CDT. These entries are effective without an IPL.

RACF authorization checking processes the dynamic CDT as a logical extension of the static CDT. The following tasks are used for adding and changing dynamic classes:

- ▶ Use the **RDEFINE** and **RALTER** commands to define and modify attributes of CDT class profiles.
- ▶ You use the **SETROPTS RACLIST(CDT)** and **SETROPTS RACLIST(CDT) REFRESH** commands to build entries in the dynamic CDT.

These commands effectively transform CDT profiles into RACF classes. The names of RACF classes created in this way (dynamic classes) can be used in RACF commands and the **RACROUTE** macro, just as you would use any other RACF class name.

When adding entries, the choice is to share a position ID with an existing entry or creating a unique position ID. When a **POSIT** value is shared between two or more classes, certain RACF processing options are controlled in the same manner (simultaneously) for all classes with the shared **POSIT** value.

For example in Figure 3-81, either one of the following commands to refresh **RACLISTed** profiles in both the **HORSES8** and **PONIES8** classes will perform both actions as they both have the same **POSIT** value.

```
SETROPTS RACLIST(HORSES8) REFRESH or  
SETROPTS RACLIST(PONIES8) REFRESH
```

Figure 3-81 SETROPTS for CDT profile

### Report on installation defined classes

The Class Descriptor report from **DSMON** includes these dynamic classes; see “The RACF class descriptor table report” on page 109. To list CDT profiles in your system, use the **RACF SEARCH** command, as shown in Figure 3-82 on page 117.

```
SEARCH CLAS(CDT)
```

```
@GILL  
MYCLASS  
MYCLAS8  
NEWCLASS  
WBEM
```

Figure 3-82 CDT profiles listed

Figure 3-83 shows program protection by RACF.

### 3.61 RACF and protecting the program

- Security modes for PROGRAM class
- Program library protection
- Program access to SERVAUTH
- Protection explanation for programs and PADS
- Execute–controlled library processing
- Program signing

Figure 3-83 Program protection by RACF

#### Program security modes

RACF provides a range of functions for controlling programs, access to program libraries, and access to data sets by programs. Program control provides the following functions:

- ▶ Simple controls to restrict the ability to execute specified programs by granting users either READ or NONE access through the PROGRAM class, and (when necessary) READ access to the DATASET profile that protects the load library that contains the program.
- ▶ More complex controls that can prevent users from copying sensitive programs or viewing the contents of such programs by granting the users either EXECUTE or NONE access through the PROGRAM class, or (in some cases) EXECUTE to the DATASET profile that protects the library that contains the program. Programs controlled in this way are referred to as *execute-controlled programs*.
- ▶ Use of the BPX.DAEMON profile within the FACILITY class in the z/OS UNIX environment.
- ▶ Program access to data sets (PADS) to allow users to have more access to data sets than they would otherwise have while running specified programs that provide restricted access to the data.
- ▶ Restricting program access to certain TCP/IP addresses through access to SERVAUTH resources.

By defining programs in the PROGRAM class, you indicate that you place some amount of trust in their behavior. Although the level of trust can vary, these programs are trusted more than programs created by general users of the system. An environment in which someone has run a program not defined in the PROGRAM class is considered a *dirty, unsafe, or uncontrolled* environment.

A clean environment is one in which only programs defined in the PROGRAM class have run. For the functions listed above, except for simple controls, RACF requires a clean environment.

Following are the program security modes:

- ▶ BASIC is the default.
- ▶ ENHANCED offers more protection but imposes additional overhead in setting up the program profiles that control program protection. It also includes further restricts the environment in which users can make use of PADS, program access to SERVAUTH resources, and execute-controlled programs.
- ▶ ENHANCED WARNING is an ENHANCED state but only operating in warning mode. It can aid the migration from BASIC to ENHANCED.

You can specify the mode through the IRR.PGMSECURITY profile in the FACILITY class. Define the profile and specify the APPLDATA operand as:

- ▶ “BASIC” for RACF to operate in BASIC program security mode.
- ▶ “ENHANCED” for RACF to operate in ENHANCED program security mode.
- ▶ Empty, or any value other than “BASIC” or “ENHANCED”, for RACF to operate in ENHANCED-WARNING program security mode.

If you do not define this profile, RACF operates in BASIC program security mode.

## Program library protection

In z/OS, program load libraries can be classified as:

- ▶ Public load libraries in the system link list concatenation (see LNKLISTxx member in SYS1.PARMLIB). Typically, users do not need explicit access to any data set profiles protecting individual libraries in the linklist concatenation if they allow the system to load the module to be executed.
- ▶ Private load libraries accessed by an explicit DD statement in batch JCL (such as JOBLIB or STEPLIB). This also means, if the load library is listed in the system link list but used explicitly with a JCL DD statement, it is deemed here as a private load library. Users will need explicit access to the data set profile protecting the library referenced by the JOBLIB or STEPLIB DD statement.

To restrict a user’s ability to run programs, you might need to protect the program library so the user cannot read from it. In some cases, you do not need to provide special protection for the program library, other than ensuring that general users cannot update it.

To restrict a user or a group from executing a particular program, you can define a profile for that program in the PROGRAM class and issue the **PERMIT** command to specify an access level of NONE for the program. You can then grant access to those users who actually need to execute the program READ access to the profile protecting the program.

To restrict a user or group from reading or copying a program but allow execution of a program, you can grant the user or group EXECUTE authority in the access list for the program library.

**Guideline:** In general, grant READ access rather than EXECUTE unless you have a strong need to prevent users from viewing the contents of a program library. Using EXECUTE requires that you keep the users' program execution environments clean, and requires more administrative effort and restrictions on how the users can access programs from the libraries.

## Program access to SERVAUTH

In this case, a number of network security zones would have been defined; they would be protected by profiles within SERVAUTH. So, user ID access to SRVAUTH resources could only occur when executing a specified program or command. Permission would occur via the WHEN(PROGRAM(*program-name*)), ID, and ACCESS parameters on the PERMIT command, as shown in Figure 3-84.

```
PERMIT resource-name CLASS(SERVAUTH) ID(user or group or *) ACCESS(READ)
WHEN(PROGRAM(program-name))
```

Figure 3-84 PERMIT command outline for the SERVAUTH class

## Protection explanation for programs and PADS

When a user ID makes a request to load (execute) a program, the following series of events occurs:

- ▶ A check occurs to see if program control is active. This is initially performed by a SETROPTS WHEN(PROGRAM) command.
- ▶ Assuming program control is active, RACF checks if this program is protected by a profile in the PROGRAM class.
- ▶ If program control is not active, RACF checks if there are any programs in this address space that are execute-protected. Also, are there any open data sets using PADS. If neither, RACF marks this as a *dirty environment* and allows the user to run the program, otherwise RACF fails the request. This is allowed because there are no programs or access to data sets that require a clean environment.
- ▶ If the program does have a profile (in the program class) protecting it and the user does not have at least EXECUTE access, the RACF fails the request.
- ▶ If ENHANCED security mode is operational further more stringent checks are made.
- ▶ If the user passes all the above and has a PADCHK attribute on the program resource profile, RACF checks if any other program access data sets are open in this address space and verifies that the user has sufficient authority.

Let us look at authorization checking for access control to data sets. The normal authorization checks apply in RACF:

- ▶ Is the UACC sufficiently high?
- ▶ Is this user's ID on the access list for the data set profile with sufficient authority?
- ▶ If a user is not granted access via normal authorization checks and program control is active, further checks are made. RACF authorizes the user to open the program-accessed data set with the currently executing program if all of the following conditions are met:
  - The conditional access list contains the name of the currently running program, the name of the first program currently running in the current task (TCB), or the name of the first program currently running in a parent task, with the requested level of access or higher.

- The user's group or user ID is associated with the program name in the conditional access list.
- The current program environment (job step, or task established under TSO/E using TSOEXEC or IKJEFTSR) is controlled. In other words, it has not loaded an uncontrolled program.
- If ENHANCED program security is active, all additional requirements are met.
- If there is more than one controlled program running in the current environment (job step or task created by TSOEXEC or IKJEFTSR), all of those programs defined with the PADCHK attribute have conditional access list entries allowing them to access the data set.

### Execute-controlled library processing

Any library where a user or program tries to open and they only have EXECUTE access is deemed an execute-controlled library. The purpose here is we want the programs to be loaded and executed but not read or copied.

First we consider the open case. Typically this would be at the start of an application. Some programs may well expect the data set to be open already and rely on earlier processes to have done so. When the request to open an execute-controlled library is made if EXECUTE is the highest access authority given to the dataset, RACF deems the request a failure. But the system will not fail it and allows the open to proceed. Then it sets a flag in the data set's control block that the user only had EXECUTE access authority.

Now the user can attempt to execute the program (or the program is fetched) from the execute-controlled data set:

- ▶ RACF checks if it is a controlled environment. If not, RACF does not allow a program to be fetched into such an environment.
- ▶ If ENHANCED mode is in use, further checks are undertaken such as ensuring the initiating program in this address space has the correct attributes and thus is not an interloper.

**Note:** EXECUTE access authority has meaning only for a partitioned data set that is used as a program library. If you specify EXECUTE for any other type of data set (such as a CLIST or EXEC), effectively the user will have an access authority of NONE.

### Program signing

RACF provides the capability to check if a program has undergone any unauthorized changes. It does require the program to be built in a special way and for a number of setup steps for RACF for this capability to be operational.

**Note:** RACF supports program signing and verification only for program objects, which are modules stored as members of a partitioned data set extended (PDSE) library.

The developers of an application might want to protect against any chance of their load modules (programs) being altered. By using the process of signing their programs, they can ensure only valid, unchanged versions of the programs are being executed.

It is a reasonably complex process from the supply of the certificate, setting up specific RACF profiles and then creating the program object. In many cases this may well be done by independent software vendors (ISVs).

At the operational level after the software product has been installed containing signed programs, RACF needs to be able to verify the signed programs. In doing so, RACF also uses signed programs of its own to verify that the programs it uses are valid and unchanged.

Again, the process is non-trivial and does require use of certificates and addition of profiles. We see that for signed programs, the PROGRAM profile will have a SIGVER segment containing signature verification options. We need to consider what actions are to flow if a signed program is not able to be verified. If it is a critical business application, it would be prudent to have a tested recovery procedure to minimize the impact.

To understand this program signing process in a detailed manner, see *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

Figure 3-85 shows RACF remote sharing facility (RRSF) nodes.

### 3.62 RACF remote sharing facility

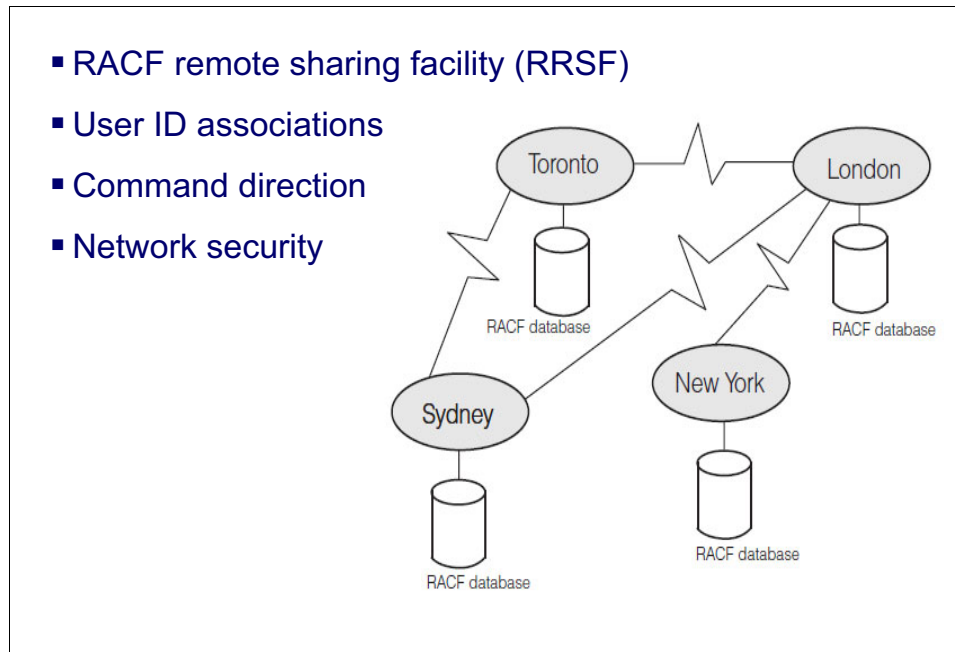


Figure 3-85 RRSF nodes

#### RACF remote sharing facility

This facility consists of a network of RRSF nodes. A node can be a z/OS system with a RACF database or a number of z/OS systems sharing a RACF database. The former is called a *single system RRSF node* and the latter is called a *multisystem RRSF node*.

In a multisystem RRSF, one of the z/OS systems is called the *main system* as this will receive the bulk of the RRSF traffic; the others are called *nonmain systems*. This is depicted in Figure 3-86 on page 122.

### 3.63 RRSF nodes and modes

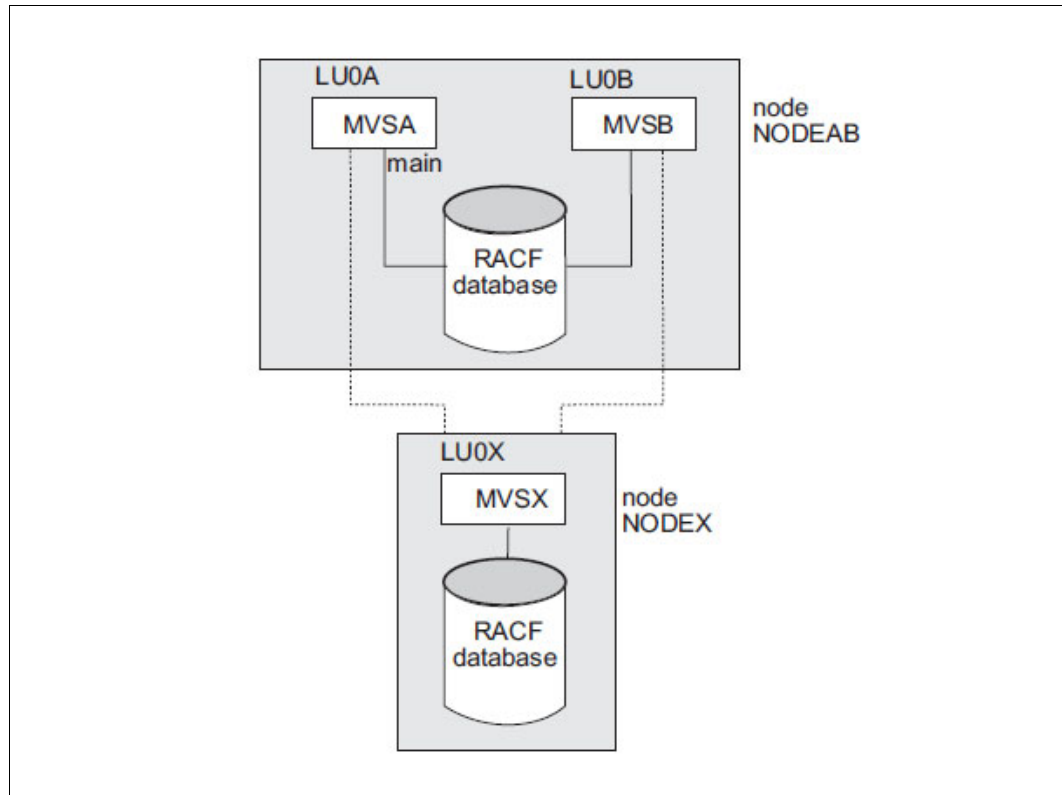


Figure 3-86 System and multisystem nodes

Within RRSF, a *local node* is a node in relation to how the description of the activity is stated. A security administrator on system MVSX will refer to NODEX as their local RRSF node and other nodes are deemed *remote nodes*. This is also shown in Figure 3-86.

Operation of the RRSF node can be in the following modes:

- Local** When in this mode it cannot communicate with other nodes in the RRSF network. It will have some limited remote sharing functions within the node. These comprise synchronized passwords for users with multiple user IDs and also direct commands to run.
- Remote** In this mode, all RRSF functions are available.

Figure 3-87 on page 123 shows RRSF user ID associations.



## 3.64 RRSF: User ID associations

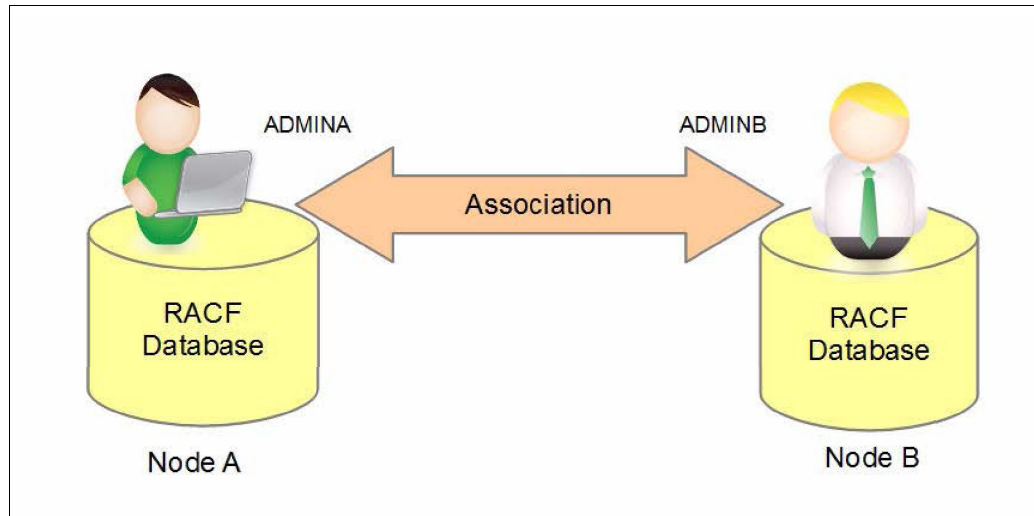


Figure 3-87 User ID associations

### User ID associations

User ID associations can be established to other RRSF nodes across the RRSF network. They can perform the following functions:

- ▶ Link or associate two user profiles on the same RRSF node or different RRSF nodes.
- ▶ Enable password association between user IDs on local and remote RRSF nodes that have the same owner and user.
- ▶ Direct commands (most but not all RACF commands) to run on other user IDs for which directing user IDs has an appropriate association.

Various types of user ID associations can exist between user IDs, such as:

**Peer** The *peer user ID association* is between two user IDs on the RRSF network. Either can issue RACF commands under the authority of the other. Password synchronization is allowed in peer associations and either user ID can delete the association.

**Managed** The managed user ID association is where one user ID is the manager. The manager user ID can direct RACF commands to managed user IDs. It is a one-way communication. However, these user IDs run these commands under their authority not of the managing user ID. Password synchronization is not permitted and any of the user IDs can delete the association.

A peer user ID association is shown in Figure 3-87.

## 3.65 RRSF: Command direction

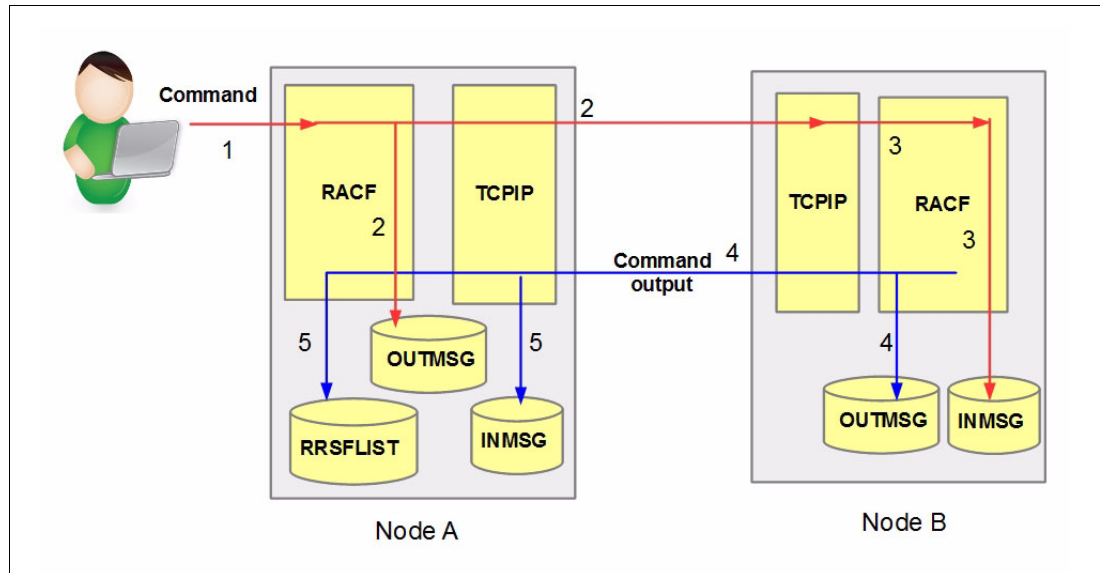


Figure 3-88 Operation flow

### Command direction

Command direction allows you to maintain remote RACF databases by using the AT keyword on specific RACF TSO commands. After a user ID association is established, RACF users can use the AT keyword to direct specific RACF TSO commands. These commands run in the RACF subsystem address space of the specified target node under the authority of the specified target user ID. Output produced by the command is captured by RRSF and written to the RRSFLIST user data set of the user ID that issued the command.

**Tip:** The naming convention for RRSFLIST data set is USERID.RRSFLIST. If ADMINA is the user ID, the RRSF data set name is ADMINA.RRSFLIST. For each user ID, RACF creates such a data set or uses a pre-allocated data set, if it exists.

Before RRSF sends the command to the target node, RACF checks whether the command issuer is authorized to run command direction for the specified target node. This process uses the RRSFDATA RACF class.

Figure 3-88 shows how the command issued by a user ID from RRSF node (Node A) to remote RRSF node (Node B) is sent. It also shows how the output is returned to the user ID that issued the command. There is no point issuing a remote command if the output is not returned showing the output from the command invocation.

Figure 3-89 on page 125 shows components of the RRSF network.

## 3.66 RRSF over TCP/IP

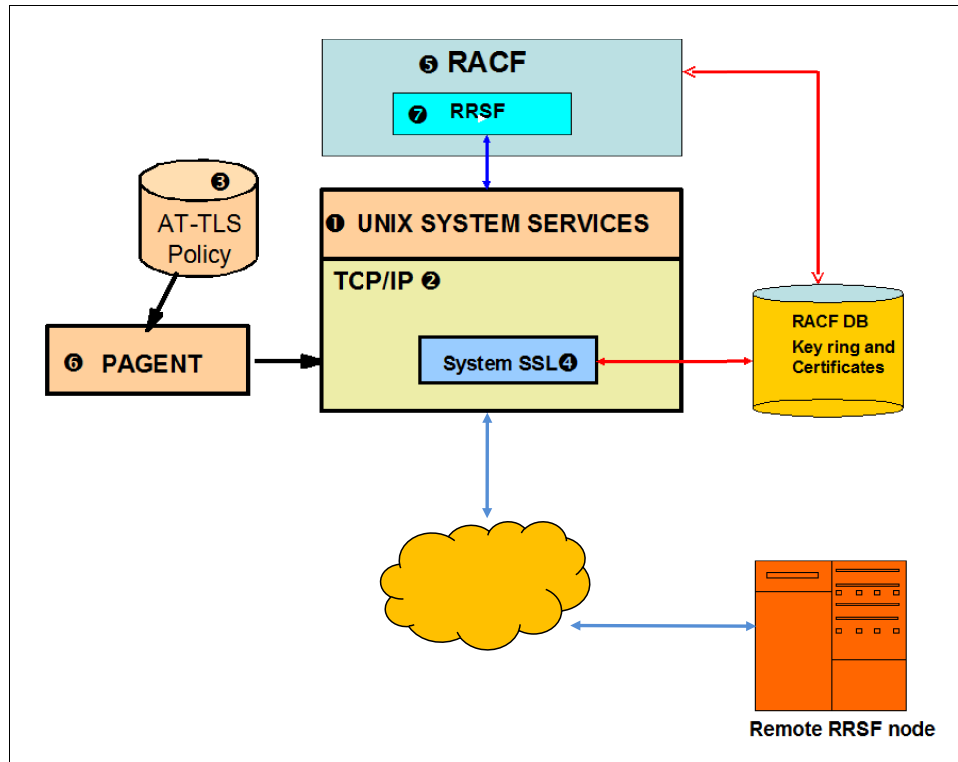


Figure 3-89 Components of the RRSF network

### RRSF over TCP/IP

RRSF (over TCP/IP) uses several z/OS system components as shown in Figure 3-89.

- 1** z/OS UNIX System Services: RRSF makes UNIX System Services socket API calls to TCP/IP.
- 2** TCP/IP: The communication vehicle for exchanging data between the RRSF nodes.
- 3** AT-TLS: The traffic between the RRSF nodes is protected by AT-TLS.
- 4** Secure Sockets Layer (SSL): AT-TLS uses SSL to authenticate the RRSF nodes.
- 5** RACF: The resources used for RRSF are protected by RACF profiles.
- 6** Policy Agent (PAGENT): The AT-TLS policy is enforced by the PAGENT.
- 7** RRSF itself.

Setting up RRSF over TCP/IP consists of the following steps:

1. Enable the RACF component of the z/OS Security Server.
2. Activate the RACF subsystem address space.
3. Requires the use of sockets, which requires UNIX System Services, to use TCP protocol. Add the following objects:
  - a. OMVS segment and UID to RACF subsystem user ID
  - b. OMVS segment and GID to the default group of RACF subsystem user ID

4. Deploy digital certificates and key rings that are used to authenticate RRSF servers to each other using TLS protocol.
5. Enable AT-TLS policy required for RRSF connections. All cryptography suites in Transport Layer Security (TLS) Protocol Version 1.2 are supported.
6. Permit the RACF subsystem user ID to the necessary resources.
7. Use the **RACF TARGET** command to complete the following steps:
  - a. Establish a socket listener on each RRSF node.
  - b. Activate the connection from both the nodes for each connection. This also supports TCP/IP V6 as well, a mixed network where links exist using different levels of TCP/IP is allowed.
  - c. Harden the **TARGET** commands in the RACF parameter library for automatic activation during system start.

The details mentioned here are explained in great detail in *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

Figure 3-90 shows RACF and interaction with other subsystems.

## 3.67 RACF and interaction with other subsystems

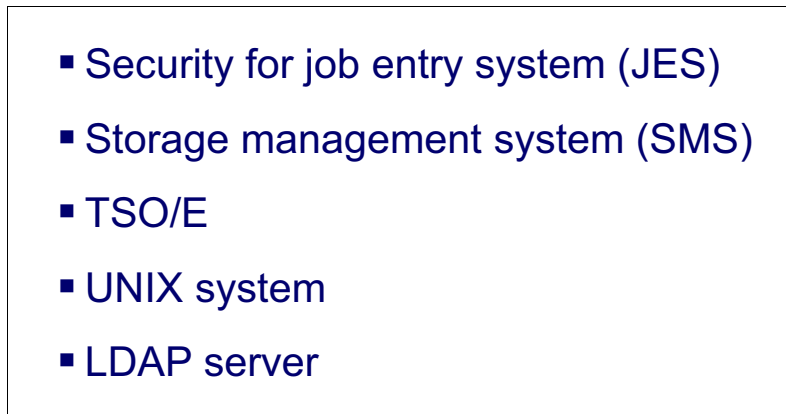


Figure 3-90 RACF and interaction with other subsystems

### Security for JES

Unless otherwise stated, JES in this context applies both to JES2 and JES3. JES is a major component of z/OS and functions as resource manager when communicating to RACF via SAF.

How can RACF provide security within the JES subsystem?

- ▶ If only authorized users are to have access to the spool input and output, JES resources would be protected by RACF.
- ▶ Restrictions can be applied to users and jobs from JES information. The restrictions would use criteria such as security labels.
- ▶ A limitation placed on users for the job names can submit and also cancel.
- ▶ Protection of SYSIN and SYSOUT if needed. SYSIN and SYSOUT could well contain sensitive information.

- ▶ Remote workstations accessing this system used to send and receive work are controlled.
- ▶ The ability for JES nodes to receive data is also controlled.
- ▶ Operators are controlled by limiting what commands they are permitted to issue.
- ▶ Additionally, operators may be restricted to what terminals they can use along with the set of commands they can issue.
- ▶ Jobs, workstations, and nodes are controlled in what commands they are allowed to submit to JES.
- ▶ Control of access to particular output devices due to that nature of their operation (for example, a printer that is used to print checks).
- ▶ For spool output, consideration could exist to print a security label on the output.
- ▶ RACF will also check if the user ID has authority to use the JOB CLASS.

The first point would be to ensure that all batch jobs have RACF identification. So, any batch job will require an explicit user ID and password on the JOB statement or it is supplied through a propagated user ID (for example, the user ID who submitted the job). This RACF command will ensure that this checking will occur. See Figure 3-91.

```
SETROPTS JES(BATCHALLRACF)
```

Figure 3-91 RACF command to ensure a check RACF identification

RACF groups would be used to assign personnel to functional groups that would cover:

- ▶ Who can issue JES commands.
- ▶ Assign users to specific terminals, so a shift operator could use the MCS console.
- ▶ Who can update system data sets owned by JES.

In Figure 3-92, there is no explicit JOB, PASSWORD, and GROUP parameter, so the RACF user ID who submitted this job will be a propagated user ID for the purposes of checking access authorization.

```
//JAMES01 JOB (1234,1234), 'IRRDUBU00', CLASS=A, MSGCLASS=X,
//          REGION=OM, TIME=1440, NOTIFY=&SYSUID
//*
```

Figure 3-92 Simple JCL

In the example shown in Figure 3-93, we have an explicit user ID and password. This pair will be used to authenticate and authorize access.

```
//HSISIQIM JOB (07F6KT,10,008,F6KT), MSGLEVEL=(1,1), MSGCLASS=H,
// NOTIFY=BOBMCC, REGION=OM, USER=TADZUSER, PASSWORD=MEA4LOU8
//*MAIN SYSTEM=SYE, CLASS=TADZ
```

Figure 3-93 Sample JES3 JOB statement

## Storage Management System

RACF does provide security options for the Storage Management System (SMS), which is a facility designed for automating and centralizing storage management. This is a part of the z/OS element called *DFSMSdfp*.

### **Supplied classes specifically for SMS**

RACF provides the following general resource classes specifically for SMS:

- MGMTCLAS** This class is designed to provide security to specific SMS management classes. This is a little confusing. The SMS management classes here are SMS constructs not RACF constructs. They contain attribute collections relating to backup and migration of data sets.
- STORCLAS** A RACF general class to protect specific SMS storage classes. A storage management class is an SMS construct, which holds attributes that relate to data set space, device, and volume.

**Note:** This does not cover SMS data classes; they do not require RACF protection.

1. Initially, we would activate these specific SMS general resources in RACF. See Figure 3-94.

```
SETROPTS CLASSACT(MGMTCLAS STORCLAS)
```

Figure 3-94 Activation of SMS-specific classes

2. Use the **RDEFINE** command to define the required RACF SMS classes to create the protective profiles. In our example, SMS has an SMS storage class called *STORE1*, so we define and ensure no ordinary user can access with a UACC(NONE). See Figure 3-95.

```
RDEFINE STORCLAS STORE1 UACC(NONE)
```

Figure 3-95 Create profile using STORCLAS

3. Now we can permit access. In this case, we allow GROUP KGN read access. See Figure 3-96.

```
PERMIT STORE1 CLASS(STORCLAS) ID(KGN) ACCESS(READ)
```

Figure 3-96 Allow access to STORE1

An alternative approach would be to define the STORE1 profile with a UACC(READ) and the use the **PERMIT** command to exclude access.

Performance issues are now considered. We have choices based on the likely number of requests to RACF for access for the SMS classes. So for SMS resource classes that you want access to by all users, create an entry in the global access checking table as depicted in Figure 3-97.

```
RDEFINE GLOBAL STORCLAS ADDMEM(STORE1/READ)  
SETROPTS GLOBAL(STORCLAS) REFRESH
```

Figure 3-97 SMS class into global access checking table

You can, of course reduce I/O to the RACF database by using the **SETROPTS RACLIST** command to keep this information in storage. This is demonstrated in Figure 3-98.

```
SETROPTS RACLIST(STORCLAS MGMTCLAS)
```

Figure 3-98 Placing SMS classes in storage

As normal, you can add or amend existing profiles for these classes and have their in-storage copies refreshed, as in Figure 3-99.

```
SETROPTS RACLIST(STORCLAS MGMTCLAS) REFRESH
```

Figure 3-99 Refreshing SMS classes

### Profile segments for SMS

RACF provides DFP segments in user, group, and data set profiles. The data within the User and Group profiles enables better control when trying to access SMS controlled data sets.

Let us see how we can explain this interaction. The DFP segment in the data set profile holds:

- ▶ An SMS data class value with attributes about the data set allocation.
- ▶ An SMS management class with attributes about migration and backup of the data set.
- ▶ An SMS storage class with attributes related to space, device, and volume.

So how can we map that to user and group profiles? In each DFP segment for these types of profiles, we have:

<b>DATAAPPL</b>	This is an identifier for a DFP data application identifier
<b>DATACLAS</b>	Default data class
<b>MGMTCLAS</b>	Default management class
<b>STORCLAS</b>	Default storage class

**Note:** RACF does not control access for DATAAPPL or DATACLAS. However, the values you specify in these fields should be defined for use on your system.

This is shown in this group list (Figure 3-100) using the LISTGRP with the DFP parameter.

```
INFORMATION FOR GROUP STOR01
SUPERIOR GROUP=SYS1          OWNER=BOBMCC      CREATED=13.212
NO INSTALLATION DATA
NO MODEL DATA SET
TERMUACC
UNIVERSAL
NO SUBGROUPS
NO USERS

DFP INFORMATION
-----
MGMTCLASS= SECURE
STORCLASS= PRIME
DATACLAS= ACCOUNTS
DATAAPPL= FINANCE
```

Figure 3-100 DFP segment listed

The ability to add DFP segment data is covered in the RACF commands that deal with users and groups. They are ADDUSER, ALTERUSER, ADDGROUP, and ALTGROUP. When adding MGMTCLAS and STORCLAS values into a DFP segment, remember they must already be defined as profiles in their matching general resource classes. There we must have profiles for SECURE and PRIME defined.

For data set profiles, the DFP segment also contains a RESOWNER field, which can supply a user ID or group of this SMS-managed data set protected by this profile. This provides the ability within RACF to have an owner who is different from the user ID who created the data set.

This is shown here with this command in Figure 3-101. We created a data set profile with a RESOWNER value.

```
ADDSD 'DESIGN.PROJ042.*' DFP(RESOWNER(ENGINE1)) UACC(NONE)
```

Figure 3-101 Create a data set with a resowner value in the DFP segment

A display of the group and the DFP segment shows the value of ENGINE1 in Figure 3-102. Do not confuse the owner of the profile DESIGN with the RESOWNER, which indicates the entity who represents the data set owner for data set allocation purposes.

```

INFORMATION FOR DATASET DESIGN.PROJ042.* (G)

LEVEL  OWNER      UNIVERSAL ACCESS  WARNING  ERASE
-----  -----  -
00     DESIGN          NONE        NO       NO

AUDITING
-----
FAILURES(READ)

NOTIFY
-----
NO USER TO BE NOTIFIED

YOUR ACCESS  CREATION GROUP  DATASET TYPE
-----  -
ALTER        SYS1            NON-VSAM

NO INSTALLATION DATA

DFP INFORMATION
-----
RESOWNER= ENGINE1

```

Figure 3-102 List of group and its DFP segment

SMS needs to be told to use RACF to acquire DFP segment data and this is performed by settings in SYS1.PARMLIB. Refer to *z/OS DFSMSdfp Storage Administration*, SC23-6860 for information about how to accomplish this.

SMS calls RACF to determine the owner of an SMS-managed data set and acquire the DFP attributes and uses them as input to the automatic class selection (ACS) routines if performing an allocation of the data set. It will also check that the owner (RESOWNER) has read access to (that is, allowed to read) the MGMTCLAS and STORCLAS RACF classes.

RACF can be set up also to control access at the field level and this can be applied to fields in the DFP segment. Figure 3-103 on page 131 shows an example for the RESOWNER field in the DFP segment.



```
SETROPTS RACLIST(FIELD)
RDEFINE FIELD DATASET.DFP.RESOWNER UACC(NONE)
SETROPTS RACLIST(FIELD) REFRESH
```

Figure 3-103 Protecting at field level in a DFP segment

Additionally RACF can be used to protect SMS resources, as the operation of SMS does require only authorized users to perform its functions. Refer to *z/OS DFSMSdfp Storage Administration, SC23-6860*, and *z/OS DFSMSdfp Storage Administration, SC23-6870*.

### **TSO/E**

For a TSO user, they need to have a user ID in the RACF database and in addition a TSO segment defined as well. We use the term TSO in this text rather than TSO/E.

**Note:** Without RACF, TSO users normally required an entry in the SYS1.UADS data set. RACF replaces that requirement. However, some user IDs (for example, IBMUSER) should be retained in SYS1.UADS for those times when RACF is inactive.

We have several areas within TSO to protect using RACF. Now that we have a TSO segment with the USER profile, we must consider protecting the general resource CLASS that they refer to:

<b>TSOPROC</b>	To protect from users modifying the defined logon procedures
<b>ACCTNUM</b>	No change allowed to the account number unless authorized to do so
<b>PERFGRP</b>	Control over the assigned performance group
<b>TSOAUTH</b>	Exercise some control over what TSO authority can be exercised

To bring this to an active state, we would issue the command shown in Figure 3-104.

```
SETROPTS CLASSACT(TSOPROC ACCTNUM PERFGRP TSOAUTH)
```

Figure 3-104 Activate TSO-specific class

As an example, let us assume we want user ID JAMES to be able to use a specific TSO logon procedure and no one else is allowed. This is a logon procedure created for one specific user ID. We add a profile LOGJAMES to the TSOPROC CLASS and ensure no other access permitted by ordinary users by having a UACC(NONE), shown here in Figure 3-105 with a refresh of the TSOPROC class in-storage.

```
RDEFINE TSOPROC LOGJAMES UACC(NONE)
PERMIT LOGJAMES CLASS(TSOPROC) ID(JAMES) ACCESS(READ)
SETROPTS RACLIST(TSOPROC) REFRESH
```

Figure 3-105 Create a profile in TSOPROC class

This should map to the TSOPROC field in the TSO segment for the user ID JAMES, as shown in Figure 3-106 on page 132, which was produced by the command LISTUSER JAMES TSO.

```
TSO INFORMATION
```

```
-----  
ACCTNUM= MVS  
PROC= LOGJAMES  
SIZE= 02096128  
MAXSIZE= 02096128  
UNIT= SYSDA  
USERDATA= 0000  
COMMAND= ISPPDF
```

Figure 3-106 Extract from LISTUSER command

Consider other matters:

- ▶ Field level access checking can be applied to fields in the TSO segment of the profile.
- ▶ As users have the ability to use the TSO **SEND** command to transmit messages, control can be exercised and the sender must have authorization to send messages.
- ▶ As TSO users usually have a strong requirement to view the JES spool, profiles in the JESSPOOL class can be used to control access. The TSO commands **OUTPUT** and **RECEIVE** are also controlled by profiles.

### UNIX System Services

RACF is used to control a number of operational aspects of the z/OS UNIX system. How do we start? Let us begin with UIDs and GIDs and continue up to protecting a zFS:

- ▶ The use of UIDs and GIDs should be very familiar to users of a UNIX system and therefore the issue of assigning values to these identifiers. Assigning such values is done using an **ALTGROUP** or **ALTUSER** when amending an existing entity. On a new user or group it will be done by the **ADDUSER** or **ADDGROUP** command. In both cases, an OMVS segment is added to the profile. In this example, we add a UID value and other OMVS segment information. See Figure 3-107.

```
ALTUSER JAMES OMVS(UID(13351) HOME('/') PROGRAM('/bin/sh'))
```

Figure 3-107 Adding an OMVS segment to an existing user

- ▶ Using the command in Figure 3-108, we are shown how the OMVS segment has been updated to hold the UID value along with the initial directory path name and the program path name.

```
LISTUSER JAMES OMVS  
...  
OMVS INFORMATION  
-----  
UID= 0000013351  
HOME= /  
PROGRAM= /bin/sh  
CPUTIMEMAX= NONE  
ASSIZEMAX= NONE  
FILEPROCMA= NONE  
PROCUSERMA= NONE  
THREADSMA= NONE  
MMAPAREMA= NONE
```

Figure 3-108 LISTUSER command to display OMVS segment

- ▶ Now a UID of 0 can perform any z/OS UNIX function and will pass all z/OS UNIX security checks. This is called *superuser authority*. This authority can be controlled by the following options:
  - The preferred method is to create resource profiles in the UNIXPRIV class. This class can be used to define profiles to grant RACF authorizations for UNIX privileges. This provides a way to granulate these commands and reduce the number of superusers needed.
  - The FACILITY class has the BPX.SUPERUSER resource.
  - The least preferred option is assigning multiple user IDs in RACF with UID(0).
- ▶ RACF can assign user limits for resource consumption within z/OS UNIX. This could be used to provide certain users with higher resource limits rather than assign superuser authority. These resource limits are stored in the OMVS segment of the user profile. Some can be seen in Figure 3-108 on page 132.
- ▶ The other part of the pairing is the GID. RACF is able to define a GID value for an existing group. It is stored in the OMVS segment belonging to the affected group profile. The following command in Figure 3-109 shows how a GID is assigned to the MFG group.

```
ALTUSER MFG OMVS(GID(4311))
```

Figure 3-109 Assigning GID to an existing group

And in Figure 3-110, we see the contents of the OMVS segment using the **LISTGRP** command.

```
LISTGRP MFG OMVS
...
OMVS INFORMATION
-----
GID= 0000004311
```

Figure 3-110 List contents of OMVS segment from group profile

**Note:** It is recommended that the same GID value not be assigned to more than one group. This is important because within z/OS UNIX security checks all groups with the same GID are treated the same and therefore any security difference between the groups is lost.

- ▶ RACF can be deployed to prevent the occurrence of shared UIDs and GIDs as they result in the loss of user accountability and decreased security. In order to do this RACF is used to establish a checking process, a list capability, and an automatic assignment of unique IDs:
  - Define a SHARED.IDS profile in the UNIXPRIV class and ensure it has RACLIST issued against it, as in Figure 3-111.

```
RDEFINE UNIXPRIV SHARED.IDS UAC(NONE)
SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)
```

Figure 3-111 Defining class to prevent duplicate UNIX identities

- As an exception, you can create a shared ID (for example, for UID 0) but you must use the **SHARED** parameter when adding to altering a user or group. An example for user

JAMES and group MFG is in Figure 3-112. This also applies when issuing the **PERMIT** command.

```
ALTUSER JAMES OMVS(UID(777) SHARED HOME(/accounts) PROGRAM(/bin/finance))
ALTGROUP MFG OMVS(GID(32) SHARED)
```

Figure 3-112 Using SHARED parameter

- Use the RACF command to search for specific UID or GIDs, as the example in Figure 3-113 shows.

```
SEARCH CLASS(USER) UID(23)
SEARCH CLASS(GROUP) GID(11)
```

Figure 3-113 SEARCH command to locate specific UNIX identifiers

- RACF will automatically generate and assign a unique UID for each user and a unique GID for a group. This can be done in several ways:
  - An explicit action to assign a unique UNIX identifier is performed by the use of OMVS(AUTOUID) and OMVS(AUTOGUID) parameters in RACF commands for users and groups respectively as illustrated in Figure 3-114 along with the subsequent message showing the new UID value being assigned.

```
ALTUSER JAMES OMVS(AUTOUID)
...User JAMES was assigned an OMVS UID value of 777.
```

Figure 3-114 Request for a UID value

- In the case where users are requesting z/OS UNIX services and do not have an OMVS segment, when their user security environment is being built, RACF can assign a unique UNIX identity. This does require a reasonable amount of setup; references to this are in *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.
- Resource names within the UNIXPRIV class are associated with z/OS UNIX privileges. By defining profiles in the UNIXPRIV class, you can protect these resources by requiring a RACF authorization to allow the grant of such z/OS privileges. In addition, this gives flexibility to the security administrator, who may allocate such z/OS UNIX privileges as needed without having to resort to assign superuser authority.
  - Essentially you create these resource profiles within the UNIXPRIV class and then map groups and users to them so they only have required privileges. As an example, consider how we can give certain users or group the ability to change the owner ID or group ID associated with a file in z/OS UNIX. The example shown in Figure 3-115 authorizes users associated with group MFG to transfer ownership of any file.

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHOWN UACC(NONE)
PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV) ID(MFG) ACCESS(READ)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

Figure 3-115 Users in group MFG can now change file ownership

Figure 3-116 on page 135 describes authentication.

## 3.68 RACF and digital certificates - the authentication problem

- Strong Authentication in a distributed environment requires cryptography
- Today's trend to authentication standardization
  - Using symmetric algorithms (shared secret keys): MIT Kerberos V5
    - widely adopted
    - practically constrained to corporate networks
  - Using asymmetric algorithms ( public key cryptography) : Digital Certificates
    - very well fitted for world wide communications
    - Standards in progress, interoperability among vendors to be established require a PKI

Figure 3-116 Authentication

### Authentication

*Authentication* is one of the primary requirements to establish trust in e-business transactions. The industry is looking for strong authentication and for standardization of the authentication mechanisms. Strong authentication uses cryptography. Two prevalently mechanisms exist today for strong authentication in a distributed environment. They differ by the kind of cryptographic algorithms that they use, which is also their domain of application.

### Cryptography

Security in communications over a non-secure network requires the use of *cryptographic procedures*. If you send data in the clear over a network that is not completely under your control from the receiver to the sender, you cannot assure the following security functions:

- ▶ Privacy: Anyone who is able to intercept your data might be able to read it.
- ▶ Integrity: An intermediary might be able to alter your data.
- ▶ Accountability or non-repudiation: It might be impossible to determine the originator of a message with confidence, and the person who sent the message can disclaim being the originator.

Security functions such as identification and authentication are also impacted because if authentication data such as passwords is sent without integrity and privacy, they can be intercepted in transit between sender and receiver, making the authentication compromised and worthless.

To ensure privacy, integrity, and accountability in non-secure networks, cryptographic procedures need to be used.

### Symmetric encryption algorithms

An *encryption algorithm* is called *symmetric* because the same key that is used to encrypt the data is also used to decrypt the data and to recover the plain text. The cipher and decipher processes are usually mathematically complex non-linear permutations.

Most symmetric ciphers that are used are *block ciphers*, which operate on a fixed number of characters at a time.

With these ciphers, it can be assumed that a brute-force attack is the only means of breaking the cipher. Therefore, the work factor depends on the length of the key. If the key length is  $n$  bits, the work factor is proportional to  $2^{(n-1)}$ .

## Asymmetric encryption algorithms

An *encryption algorithm* is called *asymmetric* because the key that is used to encrypt the data cannot be used to decrypt the data. A different key is needed to recover the plain text. This key pair is called a *public* key and a *private* key. If the public key is used to encrypt the data, the private key must be used to recover the plain text. If data is encrypted with the private key, it can only be decrypted with the public key.

Asymmetric encryption algorithms, commonly called *Public Key Cryptosystems*, are based on mathematical algorithms. The basic idea is to find a mathematical problem that is very hard to solve.

There are four asymmetric public key algorithms in use today:

- ▶ Rivest-Shamir-Adleman (RSA)
- ▶ Elliptic Curve Digital Signature Algorithm (ECDSA)
- ▶ Digital Signature Algorithm (DSA)
- ▶ Diffie-Hellman key agreement protocol

**Note:** Diffie-Hellman keys cannot be stored RACF but can be retrieved from a PKCS #11 token. PKCS means Public Key Cryptographic Standard.

## Digital signatures

*Digital signatures* are an extension to data integrity. While data integrity only ensures that the data received is identical to the data sent, digital signatures go a step further. Digital signatures provide non-repudiation, which means that the sender of a message (or the signer of a document) cannot deny authorship, similar to signatures on paper.

## Digital certificates

The application of public-key technology requires the user of a public key to be confident that the public key belongs to the correct remote person or system with which an encryption or *digital signature* mechanism is used. This confidence is obtained by using public-key certificates. A digital certificate is analogous to a passport: the passport certifies the bearer's identity, address, and citizenship. The concepts behind passports and other identification documents, such as drivers licenses, are very similar to those that are used for digital certificates.

Identification documents are issued by a trusted authority, such as the government passport office or Department of Motor Vehicles. A passport is not issued unless the person who requests it can prove identity and citizenship to the authority. Specialized equipment is used in the creation of passports to make it very difficult to alter the information in it or to forge a passport altogether. Other authorities, for example the border police in other countries, can verify a passport's authenticity. If they trust the authority that issued the document, the information contained in it is accepted as true.

A digital certificate serves two purposes:

- ▶ It establishes the owner's identity
- ▶ It makes the owner's public key available

Similar to a passport, a certificate must be issued by a trusted authority, a certificate authority (CA) and, similar to a passport, it is issued only for a limited time. When its expiration date has passed, it must be replaced.

The digital signature of the certificate authority serves the same purpose as the special measures taken for the security of passports, such as laminating pages with plastic material, which allows others to verify the authenticity of the certificate. Using the public key of the certificate authority, the MIC can be decrypted. The message digest can be re-created. If it is identical to the decrypted MIC, the certificate is authentic.

Trust is a very important concept in passports as well as in digital certificates. In the same way as, for example, a passport that is issued by some governments, even if recognized to be authentic, might not be trusted by US authorities. Therefore, each organization or user must determine which certificate authorities can be accepted as trustworthy.

### **Digital certificate formats**

Digital certificates are built according to the X.509 standard published by the International Telecommunication Union (ITU). They can come in a range of formats and are handled by RACF in different ways:

#### **Single binary certificate**

This is a binary structure and encoded using Distinguished Encoding Rules (DER). This is independent of ASCII or EBCDIC and is not aligned to a platform. It can be looked at with an editor, as certificates do not contain the private key. Do remember to use a binary mode when transferring it.

#### **PKCS #7 binary certificate package**

This a package holding one or more certificates, which could be an entire certificate chain. It can be used as a vehicle to distribute certificates. The package is not signed nor is it encrypted; it does not contain private keys.

#### **PKCS #12 binary certificate package**

The form of this that RACF supports is the password encrypted package, which does include the private key. It can be used to migrate the end entity certificate and its private key along with the certificate signing chain from one site to another. A CA can also use it to distribute a certificate chain and private key, where this CA generates the private key.

#### **Base-64 encoded certificates**

The base64 algorithm is used to convert binary data into a very simplified text, which should undergo no changes when transmitted. You can move it by cutting and pasting because the encoding is very simple and they are simply text as shown in Figure 3-117 on page 138.

```

-----BEGIN CERTIFICATE-----
MIICYzCCAcygAwIBAgIBADANBgkqhkiG9w0BAQUFADAuMQswCQYDVQQGEwJVUzEM
MAoGA1UEChMDSUJNMREwDwYDVQQLEwhMb2NhbCBDQTAeFw050TEyMjIwNTAwMDBa
Fw0wMDEyMjMwNDU5NTIaMC4xCzAJBgNVBAYTA1VTMQwwCgYDVQQKEwNJK0xETAP
BgNVBAsTCExvY2FsIENBMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD2bZEo
7xGaX2/0GHkrNFZvlxBou9v1Jmt/PDiTMPve8r9FeJAQ0QdvFST/0JPQYD20rH0b
imdDLgNdNynmyRoS2S/IIInfpmf69iyc2G0TPyRvmHIi0ZbdCd+YBHQi1adkj17ND
cWj6S14tVurFX73zx0sNoMS79q3tuXKrDsxeuWIDAQABO4GQMIGNMEsGCVUdDwGG
+EIBDQQ+EzxHZW51cmF0ZWQgYnkgdGh1IFN1Y3VyZVdheSBTZWN1cm10eSBTZXJ2
ZXIgzM9yIE9TLzM5MCAoUkFDRikwDgYDVR0PAQH/BAQDAgAGMA8GA1UdEwEB/wQF
MAMBAf8wHQYDVR00BBYEFJ3+ocRyCTJw067dLSwr/na1x6YMA0GCSqGSIb3DQEB
BQUAA4GBAMAQzt+zaJ1GU77yz1r8iIMBXgdQrwsZZWJo5exnAucJAEYQZm0fyLiM
D6oYq+ZnfVm0n8G/Y79q8nhwvuxpY0nRSAXFp6xSkrIOeZtJMY1h00LKp/JX3Ng1
svZ2agE126JHsQ0bhzn5TKsYfbwfTfjdWAGy6Vf1nYi/r0+ryMO
-----END CERTIFICATE-----

```

Figure 3-117 Example of base64 and its layout

Figure 3-118 shows an overview of a digital certificate.

### 3.69 Overview of digital certificates

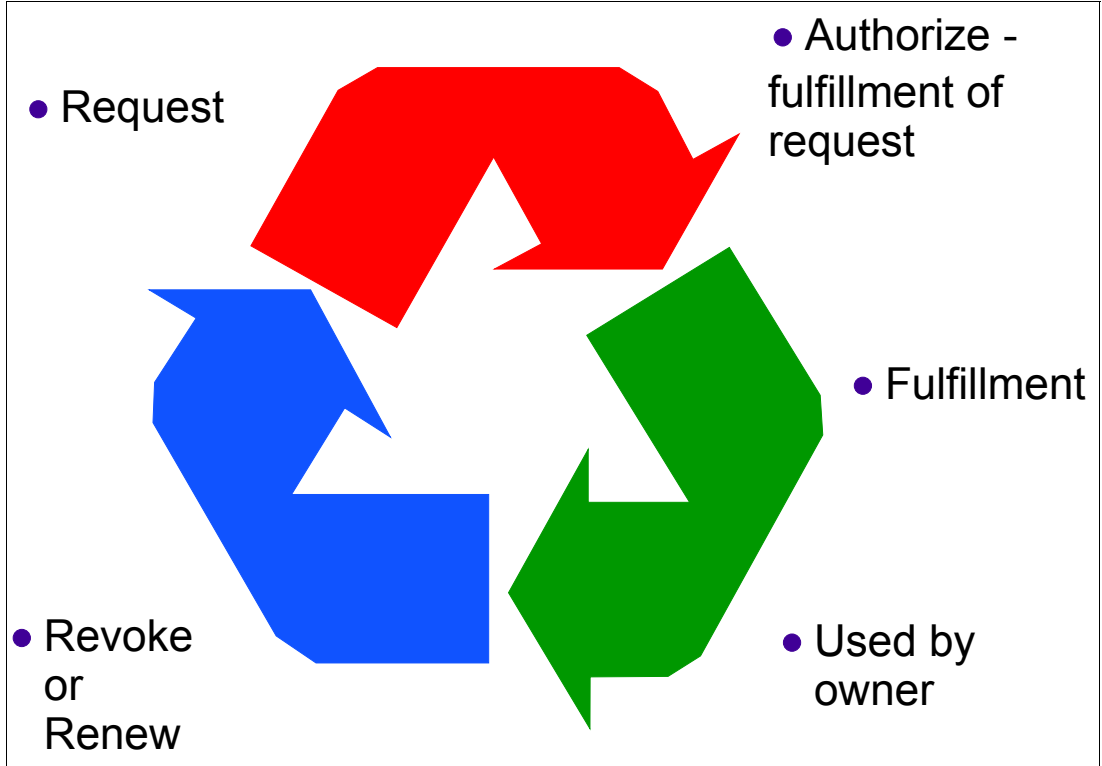


Figure 3-118 Overview of digital certificate

When compared to other known means of strong authentication, digital certificates (and the underlying public key cryptography algorithm) appear to be probably the best solution to the



current authentication and encryption problem involving a very large population of users over a non-secure network such as the Internet.

However, the use of digital certificates, over the Internet or in an intranet environment, requires a supporting public key infrastructure (PKI), which is the set of services, tools, and policies that enable the use of public key cryptography and management of keys and certificates in a PKI domain. The certificates and the associated key pairs are expected to have a lifecycle as described in Figure 3-118 on page 138.

## Life cycle stages of a digital certificate

The lifecycle stages of a digital certificate are as follows:

### 1. Request phase

First there is the generation of an asymmetric algorithm key pair. Usually, this generation is performed locally by the entity that requests the certificate, although some PKI implementations can perform the key generation on behalf of the certificate requestor. In that case, the PKI must be able to deliver the generated private key securely to the certificate's owner.

The generation of a certificate request usually contains the following elements, which are signed by the requester's private key as a proof of origin of the certificate request:

- The distinct name of the requester
- The value of the public key
- Miscellaneous additional fields

### 2. Authorize fulfillment of request and fulfillment phases

Generation and signature of the certificate itself by a CA. The CA, which can be assisted by a Registration Authority (RA) in charge of verifying the validity and integrity of the certificate request and approving it, is the pivotal entity in a PKI. The CA is in charge of digitally signing the certificate (that is, vouching for the binding of the public key value to the name of the certificate's owner), thus making the certificate usable for strong authentication or other cryptographic processes. Strictly speaking, the RA provides for the Authorize fulfillment of request phase, and the CA performs the Fulfillment.

These two phases heavily engage the responsibility, and in many cases the liability, of the CA, which is to use the proper administrative procedures and highly secure technologies to ensure the integrity of its digital signature and of the signed contents.

Note that a certificate, as delivered by a CA, is granted a validity period determined by the CA policy. Usually, a user certificate is valid for one year.

### 3. Used by owner phase

The certificate can now be used by the owner for authentication, which works if the requester can demonstrate possession of the corresponding private key or any other cryptographic process where the value of one's public key is required. The recipient of a certificate must have the public key of the CA that signed this certificate.

This public key itself is delivered in a CA's certificate because, by the PKI principle, a CA is the only entity delivering certificates in a PKI domain.

### 4. Revocation or renewal phase

The validity of a certificate can be denied in two ways:

- The validity period of the certificate is over
- The certificate is part of a certificate revocation list (CRL), which is issued by the CA that initially issued the certificate, usually on request from the certificate's owner

The owner can request the revocation of a certificate for many reasons, including:

- Change of name of the owner
- Change of association between the owner and the CA (for example, when an employee leaves a company that is its own CA)
- Compromise or suspected compromise of the corresponding private key

The entity receiving a certificate checks for its expiration based on the receiver's local time and date. Verifying that a certificate is not part of a certificate revocation list requires the receiving entity to fetch the CRL from its repository, which usually is an LDAP directory (although some PKI implementations provide access to a CRL through the HTTP protocol). A certificate becomes part of a CRL at the completion of the Revocation phase.

During its normal lifecycle, a certificate is set to expire after a certain validity period that is indicated in the certificate at its creation. The supporting PKI must then provide a way to renew the certificate, keeping the same certificate but with a new validity period and a new certificate serial number. This is the renewal phase.

Figure 3-119 shows the use of certificates in z/OS.

## 3.70 Certificate use in z/OS

- Using certificates on z/OS
- Client logon with certificates
- Management of certificates by RACF

Figure 3-119 Certificate use in z/OS

### Using certificates with client/server applications on z/OS

To create a secure certificate environment on z/OS, we need a mix of entities:

- ▶ The z/OS HTTP Server is needed for transporting the data in encrypted form
- ▶ A middleware application that supports Application Transparent Transport Layer Security (AT-TLS) or the older Secure Sockets Layer (SSL). WebSphere Message Broker is such an example.
- ▶ Secure certificate management functions of RACF

But how does a certificate on z/OS in a client/server application on z/OS, where z/OS can be either client or server, provide for a secure connection?

- ▶ Each end of the application has their own certificate with a matching private key and a list of trusted CA certificates.
- ▶ The client needs to authenticate itself to the server and the server might also want to authenticate itself to the client.
- ▶ An exchange of certificates takes place.
- ▶ Client and server separately validate the other's certificate:
  - Checks if signature is valid
  - Subject name in the certificate is correct
  - And the certificate was signed by a trusted certificate authority

- ▶ Now we need to prove that each party validly owns their certificate by proving they have the certificate's private key without sending the key over the network. This establishes proof of possession:
  - Each party encrypts a unique value known to both parties, for example, a hash of the network traffic between them. Each signs using their private key.
  - Each party uses the other party's public key (supplied with the certificate) to decrypt and check if the values match. If OK, then proof has been established that both parties are who they say they are.
- ▶ The last act is to create a session key for this transaction. One party generates a random symmetric key, encrypts it with the other's public key, and sends it to the other party. They in turn use their private key to decrypt it. This random symmetric key is used to encrypt data traffic between them.

### Client logon with certificates

Comparing this to the use of user ID and password, we can say that the certificate provides the identification, and the proof of possession provides the authorization. However, RACF still wants a user ID because many processes in z/OS rely on a user ID being present. Therefore, following are the current certificate mapping choices:

- ▶ One-to-one mapping of certificate to user ID. It can be accomplished by a RACDCERT GENCERT, which generates a certificate and RACF registers the certificate to a user ID. Or you can use RACDCERT ADD to register an existing certificate to a user ID:
  - Registered certificates are stored in certificate profiles, and where the user ID resides on the same system the private key is also in the profile.
  - From an application view, these certificates can be bundled into a key ring for applications to gain access to them.
- ▶ Certificate name filtering is where the RACDCERT MAP creates filters to many certificates to one user ID. The filtering works using fields or portions of fields within the certificate such as subject or issuer's distinguished name, organization, geography, and so on. This is formally called *certificate name filtering*.
- ▶ A particular certificate extension is called *hostIdMappings*. This enables the certificate to communicate to a system with a host identity or what RACF calls a user ID. In this extension, there could be multiple entries to cater for access to multiple systems. It will be a sequence of host name and user ID value pairs:

By using this approach, you will not need to create certificate profiles or name filters.

Figure 3-120 on page 142 shows the management of certificates by RACF.

## 3.71 Management of certificates by RACF

- What can RACF manage
- Key sizes
- RACDCERT command
- DIGTCERT profiles
- Key rings
- Tokens
- ICSF considerations

Figure 3-120 Management of certificates by RACF

### What can RACF manage?

RACF, with commands and APIs, provides the following management capabilities:

- ▶ Create certificates
- ▶ Register certificates
- ▶ Store certificates
- ▶ Administer certificates and associated private keys
- ▶ Build certificate requests for signing by a CA
- ▶ Manage key rings of stored certificates

How does this accomplish all the above functions?

- ▶ The **RACDCERT** command manages certificates and key rings.
- ▶ An application invokes a callable service called R\_datalib, which is an API to what is called the Common Data Security Functions (CDSA) and all their data library functions. This is used to establish secure sessions between servers using SYSTEM SSL.
- ▶ The initACEE callable service (another API), which provides the ability to manage certificates for users authenticated by RACF.

### Key sizes

Key sizes are a rather complex mix of issues as we deal with different key lengths depending on the algorithm used to create the private keys. Also, United States government export regulations impose a maximum key size. This can be controlled by RACF and non-RACF code in z/OS.

Normally the key length is an indicator of strength of a key, strength being the ability to deny anyone from using cryptanalysis to decode any encrypted data without using the private key. With newer algorithms, a shorter key length of the elliptical curve cryptography (ECC) is possible.

As an example, an RSA 1024 bit key size is comparable to an ECC 192-bit key size.

When RACF signs a certificate, it needs to use a secure hashing algorithm. This is done to produce a shorter piece of text, which is a unique identifier for the certificate. This provides us with the ability to see if the certificate has been tampered with.

For more information, see *Size considerations for public and private keys* in *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289. There are numerous tables present there highlighting key types, key sizes, and key strengths.

As mentioned previously, there is a cap on the size of keys as mandated by the US Government export regulations. IBM provides on z/OS an unpriced optional feature called *z/OS Security Level 3* to provide strong encryption. Once installed, key sizes over 64 bits are possible.

## RACDCERT command

This is a very powerful command and has many parameters to perform a large number of functions. We try to show by example many of them here:

**LIST**                    **RACDCERT** can list the contents of certificates from your user ID, other user IDs or a site certificate, or a certificate authority certificate. The certificate to be displayed can be selected by user ID, its type (for SITE or CERTAUTH), its label, its serial number, or issuer's distinguished name. The example in Figure 3-121 shows the command to be issued looking for the certificate with a specific label, which is a SITE certificate.

```
RACDCERT LIST(LABEL('serverecc')) SITE
...
Digital certificate information for SITE:

Label: serverecc
Certificate ID: 2QiJmZmiiA0Fg6KFmaWfMfYWDg0BA
Status: TRUST
Start Date: 2011/04/22 00:00:00
End Date: 2012/04/22 23:59:59
Serial Number:
    >02<
Issuer's Name:
    >CN=ca.0=itso.C=us<
Subject's Name:
    >CN=server.0=itso.C=us<
Signing Algorithm: sha512ECDSA
Key Usage: HANDSHAKE, KEYAGREE
Key Type: NIST ECC
Key Size: 521
Private Key: YES
PKDS Label: SRVITSOECC
Ring Associations:
Ring Owner: RODOLFI
Ring:
    >serverecc<
```

Figure 3-121 List of a SITE Certificate along with RACDCERT command

- Some points to note about this certificate:
  - NIST is the US Government's National Institute of Standards and Technology.
  - It has an ECC key size of 521 bits, which is comparable to an RSA key size of 15360 bits.
  - It is stored in an ICSF PKDS, which is a specialized external VSAM data set.

**ADD** RACDCERT requires a data set in variable block (VB) format, but PDS or PDSE are not supported. It must contain a certificate or certificate package. When adding and there is an ECC private key, the Integrated Cryptographic Service Facility (ICSF) system must be up and set to handle PKCS#11 operations. If an RSA private key over 1024 bits is to be stored in the RACF database, the CP Assist for Cryptographic Function (CPACF) must be enabled. In the example shown in Figure 3-122, we are adding a certificate in base64 form.

```
RACDCERT ADD('JAMES.USERCERT.BASE64') ID(JAMES) TRUST WITHLABEL('USECERT1')
```

Figure 3-122 Adding a certificate to the RACF database

**Finding** Finding a certificate requires several approaches. If you know the user ID the certificate belongs to or if it is a CA or SITE certificate, use the **RACDCERT LIST** command. Alternatively, you can examine profiles that contain certificates or list the key rings for certificates. In the example shown in Figure 3-123, we list the profiles to obtain a list of certificates.

```
SEARCH CLASS(DIGTCERT)
...
OD8B4FEEAAD2185BF4756A9D29E17FFB.OU=Class1PublicPrimaryCertificationAuthority.O=VeriSign,Inc..C=US

00.personal-basic@thawte.com.CN=ThawtePersonalBasicCA.OU=CertificationServicesDivision.O=ThawteConsulting.L=CapeTown.SP=WesternCape.C=ZA
```

Figure 3-123 Extract of a list of profiles with certificates using the SEARCH command

**ALTER** Use the **RACDCERT ALTER** command to change the status or the label of a digital certificate for the specified user ID, certificate authority certificate or site certificate, or its trust status. The rest of data fields are fixed due to the nature of how the certificate is sued. To make this change, if the user has more than one certificate, we need to use its label in the command. If the user has only one certificate, using the user ID is sufficient. The TRUST status in a certificate indicates if a certificate is valid and its private key has not been compromised. In the example shown in Figure 3-124 on page 145, the **RACDCERT ALTER** command alters the TRUST status to TRUST. We also list a small portion of the certificate displaying the TRUST status. The user that executes this command needs to have authority to issue it. They require UPDATE access to the resource IRR.DIGTCERT.ALTER in the FACILITY class.

```
RACDCERT ALTER (LABEL('CLIENTECC')) ID(JAMES) TRUST
...
Digital certificate information for user JAMES:

Label: clientecc
Certificate ID: 2QfZ1sTW08bJg50JhZWjhYOD
Status: TRUST
Start Date: 2011/04/22 00:00:00
End Date: 2012/04/22 23:59:59
```

Figure 3-124 Alter TRUST status and list the first part of the certificate

## DELETE

The **RACDCERT** command can be used to delete user, site, and CA certificates. The user to issue the **RACDCERT DELETE** command requires UPDATE access to the IRR.DIGTCERT.ALTER profile in the FACILITY class for user certificates. For site and CA certificates, the access required will have to be CONTROL. If the certificate belongs to a key ring, it will be removed from that key ring during the delete process. The only artifact that will remain is when a certificate is in a PKCS #11 token it will not be deleted from the token because ICSF manages these tokens. The criteria to select the certificate uniquely is its label or serial number and issuer's distinguishing name. In the example shown in Figure 3-125, we use the certificate label to remove a certificate.

```
RACDCERT DELETE(LABEL('CLIENTECC'))
```

Figure 3-125 Delete a user certificate

## CHECKCERT

RACDCERT can examine a file and if one or more certificates are present, it displays the certificate information. If the user is not authorized or the certificate is not in RACF, no RACF information is displayed. The command is shown in Figure 3-126 on page 146 to see what certificates are present in the data set 'RRSF.SC75.OK.CERT'. In this figure you are told that the file has two certificates and they are not in RACF. It also states the chain is complete because we have both the user and its CA certificate.

```
RACDCERT CHECKCERT('RRSF.SC75.OK.CERT')
...
Certificate 1:

  Start Date: 2012/01/01 00:00:00
  End Date:   2018/05/16 09:45:00
  Serial Number:
    >16<
  Issuer's Name:
    >CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
  Subject's Name:
    >CN=RRSF SERVER.OU=IBM ITS0 POK.0=IBM.C=US<
  Signing Algorithm: sha1RSA
  Key Type: RSA
  Key Size: 1024

Certificate 2:

  Start Date: 2012/01/01 00:00:00
  End Date:   2018/05/14 16:30:00
  Serial Number:
    >10<
  Issuer's Name:
    >CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
  Subject's Name:
    >CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
  Signing Algorithm: sha1RSA
  Key Usage: CERTSIGN
  Key Type: RSA
  Key Size: 1024
...
Chain information:
  Chain contains 2 certificate(s), chain is complete
```

Figure 3-126 Contents of a file containing certificates

**KEY RING**

A key ring is a way to logically group a number of certificates. A certificate can be connected to more than one key ring. Each key ring is associated with one user ID. This is independent of the certificate owner.

The key ring has some restrictions of character that can be used, such as ampersand, asterisk, and the percent sign. In addition, it holds user certificates. The **RACDCERT ADDRING** command creates an empty key ring as shown in Figure 3-127.

```
RACDCERT ID(JAMES) ADDRING(ACCOUNT)
```

Figure 3-127 Create a key ring belonging to user ID James



The **RACDCERT LISTRING** command displays the key ring, as shown in Figure 3-128, where we list the newly created key ring for user ID JAMES. You will see the annotation *no certificates connected*. This is because no certificates have been added to this key ring.

```
RACDCERT LISTRING(ACCOUNT) ID(JAMES)
....
Digital ring information for user JAMES:

Ring:
  >ACCOUNT<
  *** No certificates connected ***
```

Figure 3-128 List a key ring belonging to a specific user ID.

The **RACDCERT DELRING** command removes a key ring, as in Figure 3-129.

```
RACDCERT DELRING(ACCOUNT) ID(JAMES)
```

Figure 3-129 Delete a key ring owned by user ID JAMES

By using **RACDCERT CONNECT**, we can add a certificate to the key ring. In our example in Figure 3-130, we place a certificate owned by user ID SMITHRL into a key ring owned by user ID JAMES.

```
RACDCERT CONNECT(ID(SMITHRL) LABEL('clientecc') RING(ACCOUNT)
USAGE(PERSONAL)) ID(JAMES)
```

Figure 3-130 Place a certificate into a key ring

**RACDCERT LIST** enables us to list the certificates present in a key ring. The command format is displayed in Figure 3-131.

```
RACDCERT LISTRING(ACCOUNT) ID(JAMES)
...
Digital ring information for user JAMES:

Ring:
  >ACCOUNT<
  Certificate Label Name          Cert Owner      USAGE      DEFAULT
  -----
  clientecc                      ID(SMITHRL)    PERSONAL   NO
```

Figure 3-131 Listing contents of a key ring belong to a specific user ID

**RACDCERT DELRING** allows the removal of a key ring. It does not remove certificates; it just removes their membership in a key ring as shown in Figure 3-132.

```
RACDCERT DELRING(ACCOUNT) ID(JAMES)
```

Figure 3-132 A key ring is deleted

**CERTIFICATE Move** We use the RACDCERT EXPORT function to move certificates from RACF to a sequential file. Various options are available and so to is the complexity of authority required to perform these functions.

Depending on which keyword you specify, you can export a certificate, a certificate and its CA chain, or a certificate and private key. The following formats are available when exporting:

- CERTB64** Specifies a DER encoded X.509 certificate that has been encoded using Base64.
- CERTDER** Specifies a DER encoded X.509 certificate.
- PKCS7B64** Specifies a DER encoded PKCS #7 package that has been encoded using Base64.
- PKCS7DER** Specifies a DER encoded PKCS #7 package.
- PKCS12B64** Specifies a DER encoded PKCS #12 package that has been encoded using Base64.
- PKCS12DER** Specifies a DER encoded PKCS #12 package.

An export for a PKCS #7 is to include the certificate and its CA chain. Export for PKCS #12 requires a password to encrypt the package as it includes a certificate and its private key. A CERT keyword only implies the certificate is to be exported.

We have seen DER before and B64 means BASE64 format. Transfer BASE64 files as text and the others as binary. Therefore, in our example in Figure 3-133, we export a certificate belonging to user SMITHRL and the output file is to be base64 format. We have listed it in the same figure.

```

RACDCERT EXPORT(LABEL('clientecc')) ID(SMITHRL)
  DSN('JAMES.EXPORT.CLIENT.ECC.BAS64') FORMAT(PKCS7B64)
....
-----BEGIN CERTIFICATE-----
MIICKAYJKoZIhvcNAQcCoIICgTCCAnOCAQExADALBgkqhkiG9w0BBwGgggJ1MIIC
YTCCAcOgAwIBAgIBAzAKBggqhkiG9w0BBwQYDVQGEwJ1czENMAsgA1UE
ChMEaXRzbzELMAkGA1UEAxMCY2EwHhcNMTEwNDIyMDQwMDAwWhcNMTIwNDIyMDM1
OTU5WjAtMQswCQYDVQGEwJ1czENMAsgA1UEChMEaXRzbzEPMAOGA1UEAxMGY2xp
ZW50MIGbMBAGByqGSM49AgEGBSuBBAAjA4GGAAQBYPLKcA11cA0CLzQQfQ+yPyzp
bmTuak0ei0EcbacSSs8cJ4fxIMrTqghgf7nA1ZkDPZdsr1NoZQZn8xZsA3ukjvMB
JOBxcZaMMq4Svjs8YSkBOuv/nSKDe4pdQE11zhrB4Z6JSyjuULSDe6DKWiIBp01z
Zo6ppQz0SMjT01RDP+ooALeJgZQwgZEWpWYJYIZIAYb4QgENBDIWMEd1bmVYXR1
ZCBieSB0aGUgU2VjdXJpdHkgU2VydMvYIGZvciB6L09TICHSQUNGKTA0BgNVHQ8B
Af8EBAMCA4gWHQYDVR00BBYEFNoroRjD1WU6WkMEhzC0jxqMde18MB8GA1UdIwQY
MBaAFCv3RD3ytiuysq1sco2Yk2bwuRPGMAoGCCqGSM49BAMEA4GLADCBhwJCAIpw
2/Rf+onN1dHmvUab8eCA+e2/Y7zKUXRp+IInqo67t643exg4kYAG0TrXPFNAN+oa
vqVEYibaEZCj1rL/PuTAAkEhHYStTxPZ26A8qWfCU1x0UaDXRS/eS0DuIaE5ez7K
tdvshak1XSbSDVS7i0YZvYWBjfjV/KGwxo1jEc/1420LnzEA
-----END CERTIFICATE-----

```

Figure 3-133 Export a certificate to a file in base64 format

**Create CERTIFICATE** In our example, we create a certificate and then raise a certificate request. The latter entity can then be sent to be signed by a CA, returned and imported back into RACF. RACF can also perform the same CA function as well should it be required.

By using **RACDCERT GENCERT**, we can create a certificate and also the private/public keys. In our example in Figure 3-134 we are generating a user certificate. We have added the **DEBUG** parameter as first-time users of this command may appreciate its help if there are any errors in their command. We have only used a small subset of the possible parameters for this command. It would be best to take some time and review this command in *z/OS Security Server RACF Command Language Reference*, SA23-2292 and consult your security administrator about what standards are to be followed in your site.

```
RACDCERT GENCERT SUBJECTSDN( CN('acme.test.com') C('US') SP('NEW YORK')
L('FISHKILL') O('ACME TEST Corp')) WITHLABEL('FREIGHTOUT') DEBUG
```

Figure 3-134 Generate a user certificate

Now look at the certificate we created in Figure 3-134. This user certificate can be listed with the command in Figure 3-135. We see some of the default values in this list with a certificate expiry in 12 months time, the private key is an RSA 1024-bit key, and the signing algorithm is sha1RSA.

If you are communicating to other commercial entities, they might impose a minimum requirement for encryption algorithms and key sizes. One example of this is the compliance expected by the US Government to their FIPS 140-2 standard, which mandates minimum requirements. This is to ensure that encryption methods being used have not been compromised by cryptanalysis.

```
RACDCERT LIST(LABEL('FREIGHTOUT')) ID(JAMES)
....
Digital certificate information for user JAMES:

Label: FREIGHTOUT
Certificate ID: 2QbC1sLUw8PG2cXJx8jj1uTj
Status: TRUST
Start Date: 2013/08/05 00:00:00
End Date: 2014/08/05 23:59:59
Serial Number:
>00<
Issuer's Name:
    >CN=acme.test.com.O=ACME TEST Corp.L=BEACON.SP=NEW YORK.C=US<
Subject's Name:
    >CN=acme.test.com.O=ACME TEST Corp.L=BEACON.SP=NEW YORK.C=US<
Signing Algorithm: sha1RSA
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
    *** No rings associated ***
```

Figure 3-135 List a certificate built using the **RACDCERT GENCERT** command

## REQUEST

Having the certificate, we now want to create a certificate request from it as the next step. This is indicated in Figure 3-136 on page 150.

```
RACDCERT GENREQ(LABEL('FREIGHTOUT')) ID(JAMES) DSN('JAMES.FREIGHT.ONE.REQUEST')
DEBUG
```

Figure 3-136 Generate a certificate request into a z/OS data set

The generated request does not have a name. No key pair is generated during the GENREQ processing. It takes the subject's distinguished name, some of the extensions (indicated below), and the public key from the specified certificate and signed with the private key associated with the specified certificate to form the certificate request. GENREQ requires that the certificate have a private key associated with it.

Typically, these requests are sent to a certificate authority; however, they can also be imported into and signed by RACF using the GENCERT function with a *request-data-set-name*. The action will depend on the corporate standards as implemented by the security administrator.

We used the **DEBUG** parameter on the previous **RACDCERT** command. Figure 3-137 shows the output from that command.

```
IRRZSGG0 : Entering
IRRZSGG0 : Import key: Keylen=635
IRRZSGG0 : Set Digest alg: keyType=1 keySize=1024
IRRZSGG0 : Generate Signature: signMode=7 txtLen=319 sigLen=2048
IRRZSGG0 : Signature status=0x80
```

Figure 3-137 Example of using the **DEBUG** parameter showing output

Now we have a certificate request file. How it is moved will depend on how the CA process requires input. Some may well require an FTP to some location. If that is the case, do not use binary send as text. As this file is in BASE64 format, it can be simply cut and then pasted into the CA website for processing.

## DIGTCERT

Authority to use the prime administrative tool for digital certificates (RACDCERT) is controlled through resources in the FACILITY class:

- |                 |  |
|-----------------|--|
| <b>DIGTCERT</b> | Profiles in this class contain information about a certificate, the certificate itself, and the private key (if applicable). |
| <b>DIGTRING</b> | Profiles in this class contain information about key rings and the certificates in the key ring.                             |
| <b>DIGTNMAP</b> | In this class, we hold information about certificate name filters.   |
| <b>USER</b>     | In this class, we hold what information on digital certificates is associated with this user ID.                             |

The preceding overview is a very bare outline but was listed to give a general picture of where the parts are before going deeper:

- |                 |   |
|-----------------|---|
| <b>DIGTCERT</b> | Normally in RACF we are used to seeing short and reasonably intelligent names. However, with the name of a DIGTCERT profile we find a combination of the certificate's serial number and the issuer's distinguished name. It is rather lengthy and is shown in Figure 3-138 |
|-----------------|---|

on page 151 where the **SEARCH** command has listed profile names for the DIGTCERT class and we show the given name. So any character in the serial number and issuer's distinguished name (IDN) that is not valid in a profile name is replaced by the X'4A' character. In our example, it appears as the cent symbol.

**Note:** Do not enable generic profile checking for DIGTCERT directly by SETROPTS GENERIC(DIGTCERT) or indirectly by STROPTS GENERIC(\*). This is because the profile names due to their construction might have generic characters.

The profile name can be up to 246 characters. If the combination of serial number and issuer's distinguished name exceeds that, the second portion of the name comprises the first portion of the IDN, a certificate hash value, and the last portion of the IDN.

```
SEARCH CLASS(DIGTCERT)
...
0D8B4FEEAAD2185BF4756A9D29E17FFB.OU=Class1çPublicçPrimaryçCertificationçAutho
rity.0=VeriSign,çInc..C=US
```

Figure 3-138 Show a profile name in the DIGTCERT class

A field in this profile is the APPLDATA which holds the RACF user ID associated with this certificate. The UACC field is used to hold the TRUST status.

The DIGTCERT class should be placed in storage to assist such applications as WebSphere Application Server as a performance measure, as listed in Figure 3-139. If you change a certificate, do not forget to issue the **SETROPTS** command with the **REFRESH** parameter.

```
SETROPTS RACLIST(DIGTCERT) CLASSACT(DIGTCERT)
```

Figure 3-139 Activating and then creating in storage copies of certificate profiles

**Restriction:** Profiles within the DIGTCERT, DIGTRING, and DIGTNMAP are maintained automatically from the RACDCERT commands. The normal RACF commands that impact on profiles such as RDEFINE, RALTER, and RDELETE cannot be used to administer them. SEARCH FILTER and RLIST might not work as these profile names contain lowercase characters.

Who can issue RACDCERT commands? Any user ID with SPECIAL is the simplest choice. But for a more fine-grained authority, we use the IRR.DIGTCERT.*function* in the FACILITY class.

The *function* value maps to the RACF command name:

READ access to the IRR.DIGTCERT.*function* permits RACDCERT commands for themselves.

UPDATE access to the IRR.DIGTCERT.*function* permits RACDCERT commands for others.

CONTROL access to the IRR.DIGTCERT.*function* permits RACDCERT commands for SITE and CERTAUTH certificates.

Careful thought should be given about how access is given, for example consider the following situations:

- ▶ The number of users who can add certificate authorities and site certificates should be small and auditable.
- ▶ End users should be responsible for their own certificates and key rings having add, delete, and modify capabilities. By end user, we typically refer to the application owner who would have certificate requirements to maintain.
- ▶ Help desk personnel at the first point of inquiry should be able to list both certificates and rings.

RACF comes with three user IDs defined in the user profiles. These are fixed and cannot be defined. They are used to anchor specific profiles in the DIGTCERT and DIGTRING class:

<b>User ID irrcerta</b>	When you add a user certificate with <b>RACDCERT ADD</b> command and use the CERTAUTH option, they are automatically associated with this user ID irrcerta.
<b>User ID irrsitec</b>	This user ID would be associated with those user certificates added using the RACDCERT ADD with the SITE option.
<b>User ID irrmulti</b>	This user ID gets associated with the certificate name filters added with the <b>RACDCERT MAP</b> command

**Note:** These special user IDs are mostly immune to RACF commands. Even if you delete them, they are added automatically at RACF initialization. Being all lowercase using the **SEARCH** command to find them is problematical.

### ***Certificate Name filtering - DIGTNMAP***

It could be said that the number of certificates under RACF control is a non-trivial number and will continue to grow. We use name filtering as a generic way to identify the operational user ID, for example, to create a security context with a client logon using a certificate such as during an SSL authentication.

This will permit more than one user to share the same RACF user ID. For example, this RACF user ID could be a functional user ID, such as ACCOUNT or TELLER. We can use the **RACDCERT MAP** command to associate a user ID to each filter we define. Why? When the certificate arrives from a client, we can match the certificate to a certificate name filter and if we match an operational user ID is now known.

We will need to have the DIGTNMAP class active as it stores the certificate name filter (mapping) profiles and the associated user ID. Following are the name filters in use:

**Issuer's name filter** Contains a full or partial issuer's distinguished name.

**Subject's name filter** Contains a full or partial subject's distinguished name.

### ***Subject's and Issuer's name filter***

The example in Figure 3-140 on page 153 shows two subject name filters based on partial subject's distinguished names. We associate the name filters to specific user IDs WIDGETS & NYUSER.

```
RACDCERT ID(WIDGETS) MAP WITHLABEL('NY SALES REPS') TRUST
SDNFILTER('OU=Sales.OU=New York.OU=US.O=World Sales Corp')
RACDCERT ID(NYUSER) MAP WITHLABEL('NY OTHERS') TRUST
SDNFILTER('OU=New York.OU=US.O=World Sales Corp')
SETROPTS RACLIST(DIGTNMAP) REFRESH
```

Figure 3-140 Commands to create certificate name filters

If a certificate should match both filters, the filter chosen will have the most specific details. It should be noted that certificate processing will first search the DIGTCERT class for an exact match using the certificate's serial number and issuer's distinguished name. If no match occurs, the DIGTNMAP class is searched for a match on a certificate name filter.

## Key rings

A key ring is a logical way to connect together a number of certificates. The key ring has one owner and the certificates connected to the key ring do not necessarily have to belong to the key ring owner.

Typically, an application on z/OS would have specified as a configuration or customization parameter the name of a key ring. The classic example of this is the TLS/SSL handshake between server and client. The server key ring would also hold the CA certificate to validate the client certificate if client authentication is undertaken.

Now a *virtual* key ring is the set of all certificates owned by one user ID. The most typical use of this would be a client/server application, where this application validates the certificate of others and has no need for its own certificate and private key.

## DIGTRING

This is a general resource class. This class holds profiles for key rings. These profiles will have references to those certificates that are part of the key ring. A profile name is made up of:

userid.key-ring-name

This class will need to be active before key rings are used. This class can be processed by the **SETROPTS RACLIST** command for performance reasons.

The same restriction for generic profile checking for DIGTCERT also applies to the DIGTRING class.

So, how is this used? If we look at the **RACDCERT ADDRING** command, we need to specify a *ring-name* up to 237 characters (we exclude ampersand, asterisk and percent sign characters), and a user ID to be the *ring-owner*.

```
RACDCERT ADDRING(FinanceMonthlyDataRing) ID(ACCOUNT)
```

User ID ACCOUNT wants to add a key ring. The certificates that will be connected to this new ring will be shared by multiple users and this ring will represent the installation's trust policy for access and use of the FinanceMonthlyData application.

## TOKENS

A container holding a certificate and keys is called a *token*. z/OS supports PKSC #11 tokens but the tokens are managed by ICSF. However, RACF can be used to maintain specific certificate objects such as certificates, public keys, and private keys.

You can use the following RACDCERT command functions:

<b>ADDTOKEN</b>	Defines a new empty token.
<b>DELTOKEN</b>	Deletes an existing token and all its contents.
<b>LISTTOKEN</b>	Displays information about the objects contained in the token.
<b>BIND</b>	Connects a RACF certificate, its public key, and (in some cases) its private key, to an existing token.
<b>UNBIND</b>	Removes a certificate and its keys from an existing token.
<b>IMPORT</b>	Adds a certificate to RACF from an existing token.

Because other applications can use functions provided by ICSF, changes could be made to tokens without RACF becoming aware of them. Similarly, if RACF made changes to a certificate that is bound to a token, this may well not be communicated to ICSF to maintain its token information.

Therefore, the following restrictions are in place:

- ▶ Deleting, altering, or renewing a RACF certificate that is bound to a token has no effect on the equivalent token objects managed by ICSF.
- ▶ Deleting or altering a certificate object in a token has no effect on the following objects:
  - The equivalent RACF certificate.
  - The equivalent certificate objects in other tokens.

### ICSF considerations

How does ICSF interact with RACF? ICSF does several things:

- ▶ It provides APIs to the cryptographic hardware (this is the hardware cryptographic coprocessor). Using hardware to perform encryption and decryption is a far more efficient and timely process than performing the same function in software.
- ▶ It provides the best solution for storage of private keys with hardware protection.
- ▶ It makes sure that private keys are stored in an encrypted form using its master key and stored in the ICSF PKA key data set (PKDS).

RACF plays a role by providing general resources in the CSFKEYS and CSFSERV class. Also, moving non-ICSF private keys to ICSF can be accomplished by using the **RACDCERT ADD** command with certain functions.

**Note:** Because private keys stored in ICSF can be recovered in clear form, use of RACDCERT EXPORT of a certificate into a PKSC #12 form is not possible. An exception would be if the destination is z/OS and it has the same ICSF PKA master key.





# Integrated Security Services

This chapter examines the z/OS implementation of Network Authentication Services (Kerberos), Enterprise Identity Mapping (EIM), and Open Cryptographic Enhanced Plug-ins (OCEPs).

## 4.1 Introduction to Kerberos

- A distributed authentication service developed by MIT
- Currently at Version 5
- Allows user authentication over a physically untrusted network without transmitting password
- Tickets are issued by a Kerberos authentication server: both users and servers are required to have keys registered with the authentication server
- The data flows establish the session key that is used in a direct exchange
- Optionally provides data privacy

Figure 4-1 Introduction to Kerberos

### Introduction to Kerberos

Kerberos is a network authentication protocol that was developed in the 1980s by Massachusetts Institute of Technology (MIT), in cooperation with IBM and Digital Equipment Corporation. Data Encryption Standard (DES) cryptography and Advanced Encryption Standard (AES) are used to provide data privacy, especially for the sensitive data, such as a password to log in to a server.

In z/OS, this component's formal name is *Integrated Security Services Network Authentication Service for z/OS*; however, in this book, we refer to it as *Network Authentication Service for z/OS*. The GSS-APIs support the SPKM-3/LIPKEY mechanisms.

Kerberos is an encryption-based security system that provides mutual authentication between the users and the servers in a network environment. The assumed goals for this system are as follows:

- ▶ Authentication to prevent fraudulent requests/responses between users and servers that must be confidential and on groups of at least one user and one service.
- ▶ Authorization can be implemented independently from the authentication by each service that wants to provide its own authorization system. The authorization system can assume that the authentication of a user/client is reliable.
- ▶ Message confidentiality can also be used that provides assurance to a data sender that the message's content is protected from access by entities other than the context's named peer.

The Kerberos authentication is based heavily on shared secrets, which are passwords stored on the Kerberos server. Those passwords are encrypted with a symmetrical cryptographic algorithm, which is DES and AES in this case, and decrypted when needed. This fact implies that a decrypted password is accessed by the Kerberos server, which is not usually required in an authentication system that uses public key cryptography. Therefore, the servers must be placed in secure locations with physical security to prevent an attacker from stealing a password.

For a complete description of the supported Request for Comments (RFCs), see the following site:

<http://www.ietf.org>

For a list of supported RFCs, see *z/OS V2R1.0 Integrated Security Services Network Authentication Service Administration*, SC23-6786.

Figure 4-2 shows Kerberos terminology.

## 4.2 Kerberos terminology

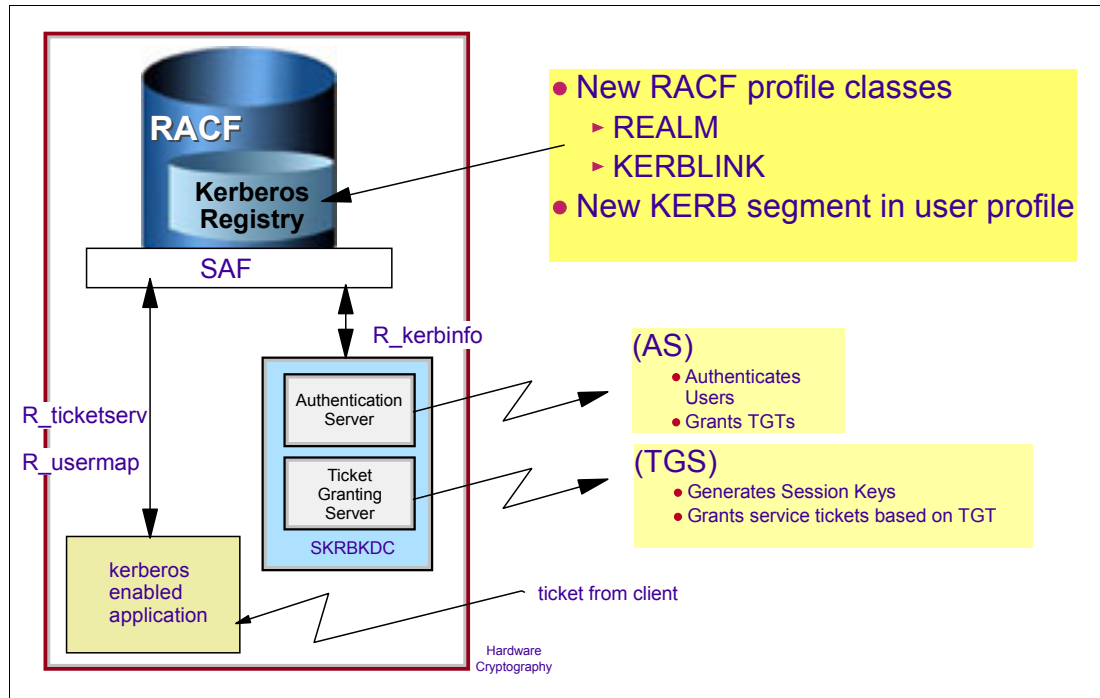


Figure 4-2 Kerberos terminology

### Kerberos terminology

Kerberos terminology includes:

- ▶ **Realm:** The Kerberos domain that is the set of entities, which authenticate using that Kerberos key distribution center (KDC).
- ▶ **Principal:** A client or an application server in a Kerberos domain.
- ▶ **Instance:** Additional distinction between principals names.
- ▶ **Kerberos name:** principal\_name.instance@realm.
- ▶ **Kerberos ticket:** The ticket is encrypted under a key that is only known to the Kerberos KDC and the end server. The ticket includes the following components:

- Client's identity
- A dynamically created session key
- A time stamp
- A lifetime for the ticket
- A service name

A ticket can be reused during its lifetime.

- ▶ **Authenticator:** Client's name and IP address as well as a time stamp. Issued with each client's request. The authenticator must be different for each request and is used for replay protection.

## 4.3 Kerberos protocol overview

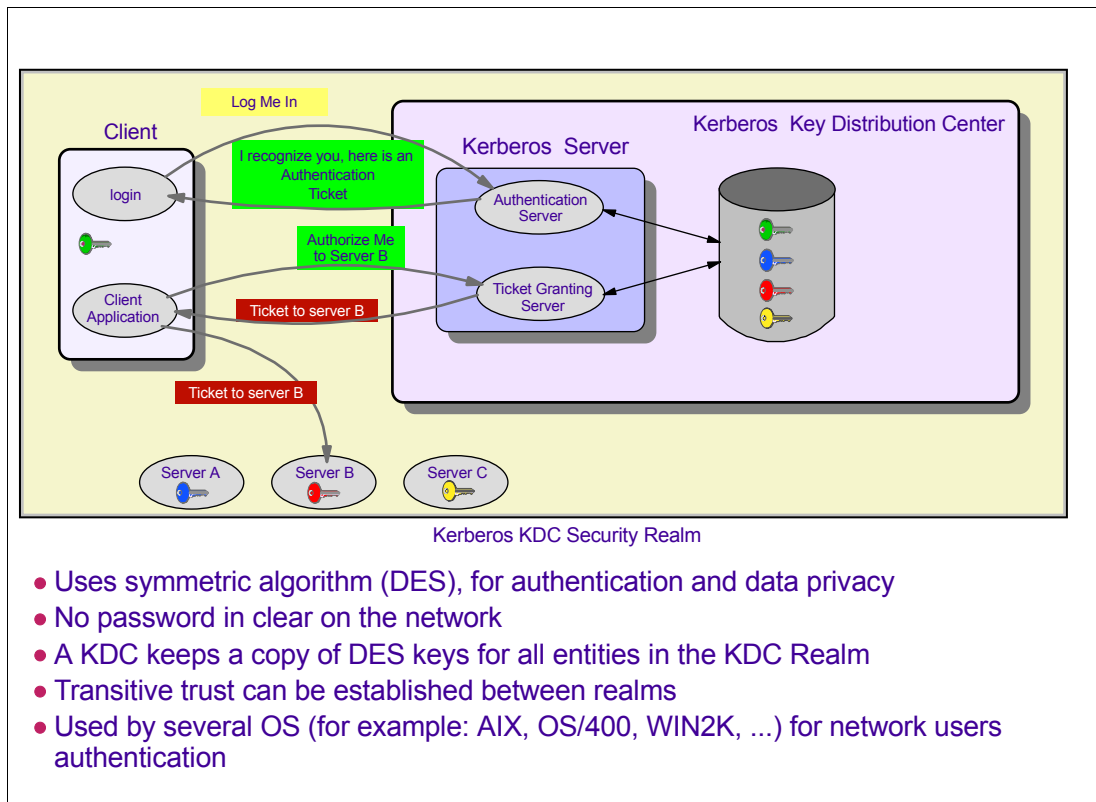


Figure 4-3 Kerberos protocol overview

### Kerberos protocol overview

The Kerberos system consists of three components:

- ▶ A client
- ▶ A server
- ▶ A trusted third party, which is also known as a *key distribution center* (KDC)

KDC interacts with both a client and server to accept the client's request to authenticate its identity, and to issue tickets to it.

The domain served by a single KDC is referred to as a *realm*. A principal identifier is used to identify each client and server in a realm. The principal name is uniquely assigned for all clients and servers by the Kerberos administrator. All principals must be known to the KDC. Kerberos realms can interoperate by establishing trust relationships, sharing secret keys, between them.

All entities in the network, clients, and servers, have their own secret symmetric key. A copy of all the keys is kept in the Kerberos Key Distribution Center. Clients' keys are derived from their password.

Kerberos is intended for corporate networks or intranets because the scalability of the protocol is directly related to the number of secret keys that can be managed in a KDC.

Although the Kerberos protocol consists of several subprotocols, three exchanges are particularly interesting to most readers. The first-phase exchange takes place between a

client and the authentication server. In this phase, a client asks the authentication server that knows the secret keys of all clients in the realm to authenticate himself and give the client a ticket (called a *ticket-granting ticket*) to be used to get a secret key, which is then shared with an application server the client wants to access.

Upon receiving the ticket-granting ticket, the client sends a request that contains the ticket-granting ticket, for a service ticket to the ticket-granting server, and waits for a service ticket to be returned. Having the session ticket (service ticket) ready, the client is allowed to communicate with the server that is providing a service that he wants to use. Optionally, the application server can perform further authentication processing against the client.

Message encoding defined in Kerberos Version 5 is described using the Abstract Syntax Notation 1 (ASN.1) syntax, in accordance with ISO standards 8824 and 8825.

In the remainder of this chapter, we describe the interactions in more detail. We use the following notations:

- ▶ **Kx**: X's symmetric encryption key
- ▶ **Kx,y**: Encryption key shared by X and Y (for example, a session key)
- ▶ **Kx{data}**: A message that contains data encrypted with X's key

Get a ticket-granting ticket is shown in Figure 4-4.

## 4.4 Get a ticket-granting ticket

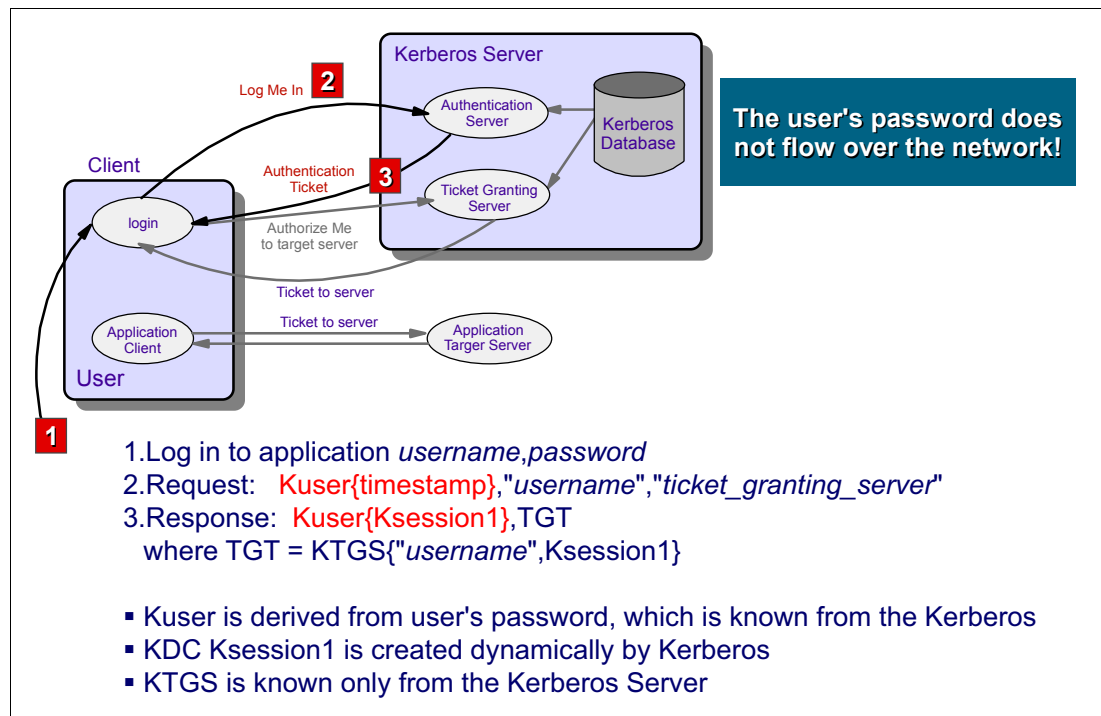


Figure 4-4 Get a ticket-granting ticket

### Phase 1: Authentication service exchange

To simplify our explanation, we use an example with a user named *Alice* (username=Alice). So, in Figure 4-4 the numbered steps become:

1. Alice enters a user name and password
2.  $K_{Alice}\{timestamp\}$ , "Alice", tgs, nonce
3.  $K_{Alice}\{K_{Alice,KDC}, nonce\}$ , TGT, where  $TGT = K_{Alice}\{ "Alice", K_{Alice,KDC} \}$

The authentication service exchange is initiated by a client when it wants to get authentication credentials for an application server but currently holds no credentials. Two messages are exchanged between the client and the Kerberos authentication server, then credentials for a ticket-granting server are given to the client. These credentials are the so-called *ticket-granting ticket*, which is used subsequently to obtain credentials for other services.

This exchange is used for other services, such as the password-changing service, as well.

**Note:** The client's secret key is used exclusively in this phase.

When you log in to a client system and enter your password, a client sends the Kerberos authentication server a message that includes a user name in plain text ("Alice"), the current time encrypted with her secret key, and the identity of the server for which the client is requesting credentials.

Upon receiving the request from the client, the authentication server looks up the client name and the service name (the ticket-granting service in this case) in the Kerberos database, and obtains an encryption key for each of them,  $K_{Alice}$  and  $K_{KDC}$ .

The authentication server then generates a response back to the client, which contains the ticket-granting ticket and a session key  $K_{Alice,KDC}$ , which is used in the subsequent secure communication between the client and KDC. The ticket-granting ticket includes the session key  $K_{Alice,KDC}$ , the identities of the server and the client, lifetime, and some other information. The authentication server then encrypts the ticket using its own key  $K_{KDC}$ . This produces a *sealed ticket*. The session key  $K_{Alice,KDC}$  is also encrypted using the client's key  $K_{Alice}$  with some other information, such as nonce.

The encrypted current time is also known as the authenticator because the receiver can assure that the sender knows the correct shared secret  $K_{Alice}$ , which is the client's encryption key derived from her password (this key is also referred to as Alice's *long-term key*), by decrypting it and validating what is inside. Because the authentication server knows Alice's secret key, it can evaluate the time decrypted from the received authenticator.

**Tip:** You might have noticed that the clocks on the client system and the KDC must be reasonably synchronized with each other. You can use a network time service to synchronize the clocks.

An authenticator is also used to help the server detect message replays.

Nonce is information used to identify a pair of the Kerberos request and response. You can use a time stamp or a random number generated by a client.

Tgs is the server's identification, which is the Kerberos ticket-granting server in this case.

Because  $K_{Alice}$  is known exclusively by Alice and the KDC, no one but Alice can extract the critical information from the response message, such as the session key  $K_{Alice,KDC}$  used in the next phase.

When the client receives the authentication server's response, it decrypts it using its secret key  $K_{Alice}$  and checks to see if the nonce matches the specific request. If the nonce matches,

the client caches the session key  $K_{\text{Alice},\text{KDC}}$  for future communications with the ticket-granting server. A ticket-granting service exchange is shown in Figure 4-5 on page 162.

## 4.5 Request a service ticket

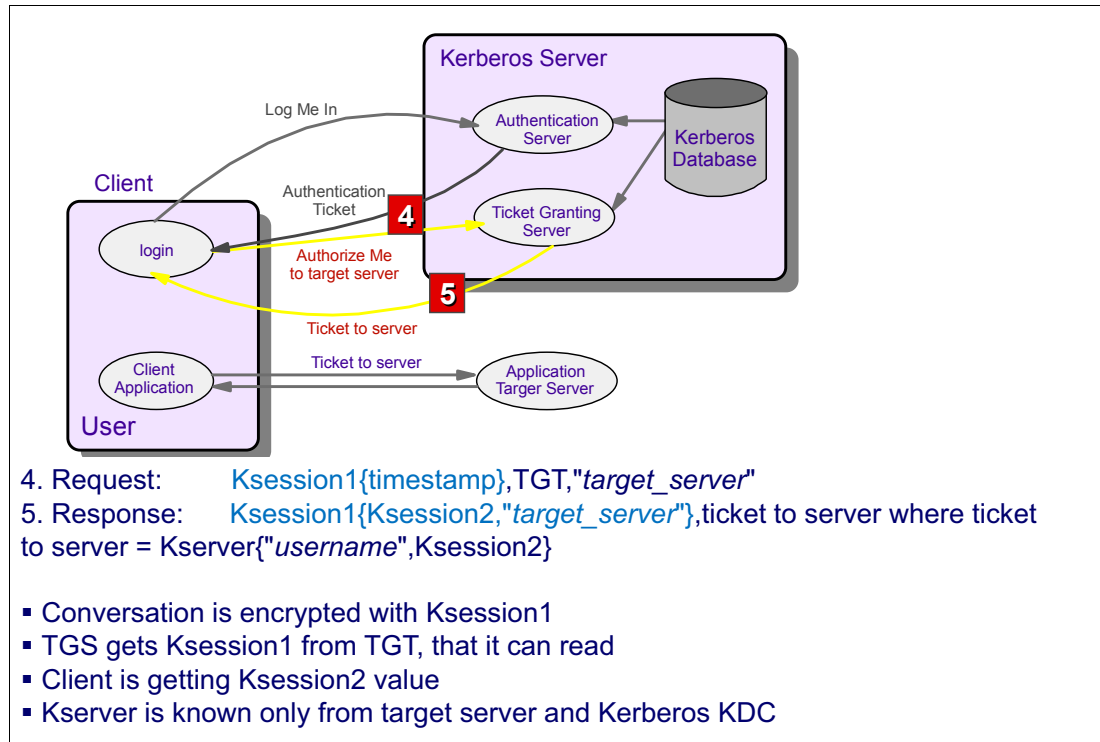


Figure 4-5 Request a service ticket

### Phase 2: Ticket-granting service exchange

Because in our example username=Alice and server=Bob, in Figure 4-5 the numbered steps become:

4.  $K_{\text{Alice}}\{\text{timestamp}\}, \text{TGT}, \text{"Bob"}, \text{nonce}$
5.  $K_{\text{Alice}, \text{KDC}}\{K_{\text{Alice}, \text{Bob}}, \text{"Bob"}, \text{nonce}\}, \text{tk\_to\_Bob}$ , where  $\text{tk\_to\_Bob} = K_{\text{Bob}}\{\text{"Alice"}, K_{\text{Alice}, \text{Bob}}\}$

When the ticket-granting server receives the message from the client, it first decipheres the sealed ticket using its encryption key  $K_{\text{KDC}}$ . From the deciphered ticket, the ticket-granting server obtains the session-key  $K_{\text{Alice}, \text{KDC}}$ . It uses this session key to decipher the authenticator.

The validity checks that performed by the ticket-granting server include verifying the following components:

- ▶ The client name and its realm in the ticket match the same fields in the authenticator.
- ▶ The address from which this message originates is found in the address field in the ticket, which specifies addresses from which the ticket can be used.
- ▶ The user-supplied checksum in the authenticator matches the contents of the request. This procedure guarantees the integrity of the message.

Finally, it checks the current time in the authenticator to make certain the message is recent. Again, this requires that all the clients and servers maintain their clocks within some prescribed tolerance.



**Important:** By checking the time stamp in the nanoseconds scale, replay attacks can be detected.

The ticket-granting server now looks up the server name from the message in the Kerberos database, and obtains the encryption key  $K_{Bob}$  for the specified service.

The ticket-granting server forms a new random session key  $K_{Alice,Bob}$  for the benefit of the client (Alice) and the server (Bob), and then creates a new ticket  $tk_{t\_to\_Bob}$  that includes:

- ▶ The session key  $K_{Alice,Bob}$
- ▶ Identities of the service and the client
- ▶ Lifetime

**Note:** The format of the ticket for a particular service is identical to one of the ticket-granting tickets.

The ticket-granting server then assembles and sends a message to the client.

Figure 4-6 shows the client/server authentication exchange.

## 4.6 Authenticate to target server

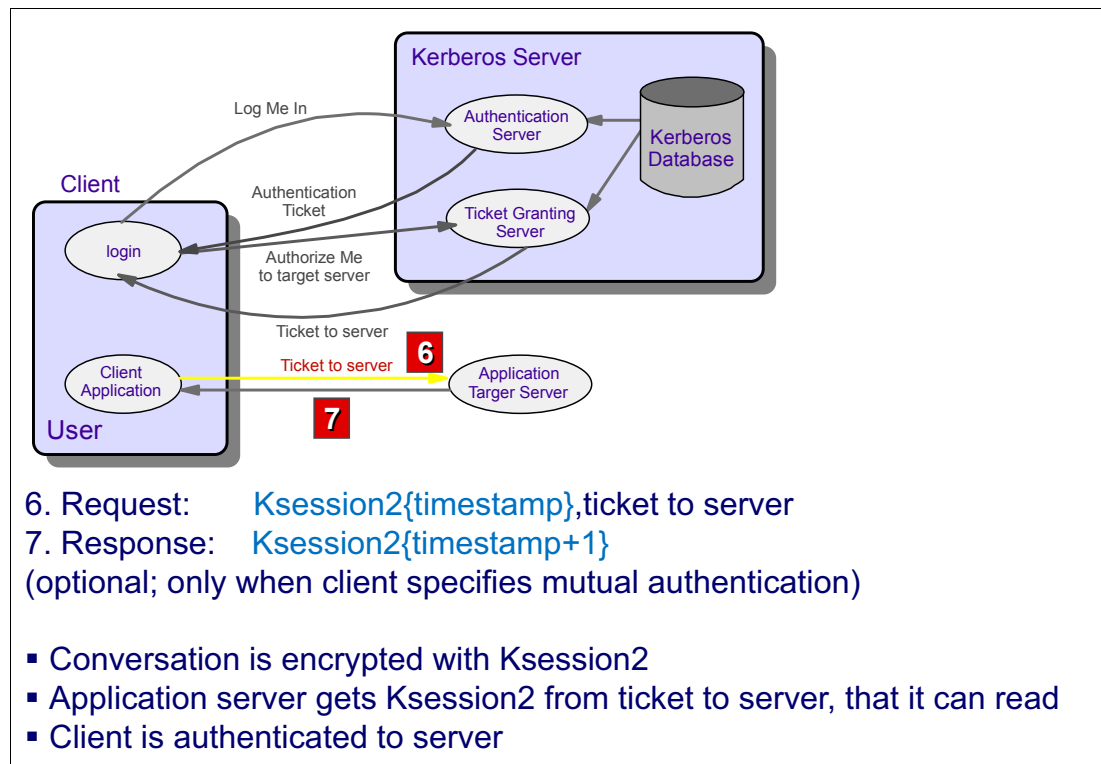


Figure 4-6 Authenticate to a target server

### Phase 3: The client/server authentication exchange

Following our example, in Figure 4-6 the numbered steps become:

6.  $K_{Alice,Bob}\{timestamp\}, tkt\_to\_Bob$

#### 7. $K_{A_{lice},KDC}\{timestamp\}$ (optional)

The client/server authentication exchange is performed by the client and the server to authenticate each other. The client has been issued credentials for the server using the authentication service or ticket-granting service exchange before the client/server exchange is initiated.

After receiving the ticket-granting server exchange response from the ticket-granting server, the client deciphers it using the ticket-granting server session key  $K_{A_{lice},KDC}$  that is exclusively known by the client and the ticket-granting server. From this message it extracts a new session key  $K_{A_{lice},Bob}$  that is shared with the server (Bob) and the client (Alice). The sealed ticket included in the response from the ticket-granting server cannot be deciphered by the client, because it is enciphered using the server/s secret key  $K_{Bob}$ .

Then the client builds an authenticator and seals it using the new session key  $K_{A_{lice},Bob}$ . Finally, it sends a message containing the sealed ticket and the authenticator to the server (Bob) to request its service.

When the server (Bob) receives this message, it first deciphers the sealed ticket using its encryption key  $K_{Bob}$ , which is kept in secret between Bob and the KDC. It then uses the new session key  $K_{A_{lice},Bob}$  contained in the ticket to validate the authenticator in the same way as the ticket-granting server does in the ticket-granting server exchange.

After the server has authenticated a client, an option exists for the client to validate the server (this procedure is called *mutual authentication*). This prevents an intruder from impersonating the server.

If mutual authentication is required by the client, the server has to send a response message back to the client. The message must contain the same time stamp value as one in the client's request message. This message is enciphered using the session key  $K_{A_{lice},Bob}$  that was passed from the client to the server.

If the response is returned, the client decrypts it using the session key  $K_{A_{lice},Bob}$  and verifies that the time stamp value matches one in the authenticator that was sent by the client in the preceding client/server exchange. If it matches, then the client is assured that the server is genuine.

When the client/server exchange has completed successfully, an encryption key is shared by the client and server and can be used for the on-going application protocol to provide data confidentiality.

Figure 4-7 on page 165 shows a Kerberos inter-realm trust relationship.

## 4.7 Kerberos inter-realm trust relationship

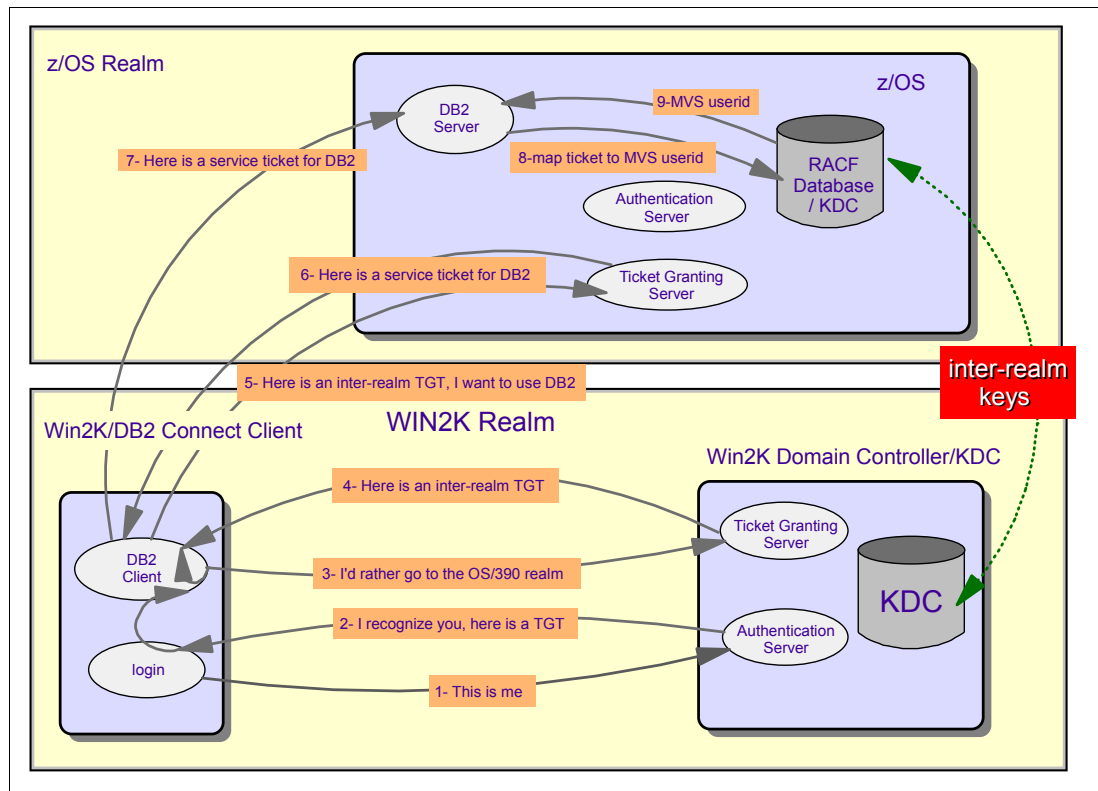


Figure 4-7 Kerberos inter-realm trust relationship

### Kerberos inter-realm trust relationship

The Kerberos protocol is designed to operate across organizational boundaries. Each organization that wants to run a Kerberos server establishes its own realm. The name of the realm in which a client is registered is part of the client's name and can be used by the application server to decide whether to honor a request.

By establishing inter-realm keys, the administrators of two realms can allow a client authenticated in one realm to use its credentials in the other realm. The exchange of *inter-realm keys* registers the ticket-granting service of each realm as a principal in the other realm. A client is then able to obtain a ticket-granting ticket for the remote realm's ticket-granting service from its local ticket-granting service. Tickets issued to a service in the remote realm indicate that the client was authenticated from another realm.

This method can be repeated to authenticate throughout an organization across multiple realms. To build a valid authentication path to a distant realm, the local realm must share an inter-realm key with the target realm or with an intermediate realm that communicates with either the target realm or with another intermediate realm.

Realms are typically organized hierarchically. Each realm shares a key with its parent and a different key with each child. If an inter-realm key is not directly shared by two realms, the hierarchical organization allows an authentication path to be easily constructed. If a hierarchical organization is not used, it might be necessary to consult some database to construct an authentication path between realms.

Although realms are typically hierarchical, intermediate realms can be bypassed to achieve cross-realm authentication through alternative authentication paths. It is important for the end-service to know which realms were transited when deciding how much faith to place in the authentication process. To facilitate this decision, a field in each ticket contains the names of the realms that were involved in authenticating the client.

Figure 4-8 shows some assumptions to Kerberos.

## 4.8 Some assumptions to Kerberos

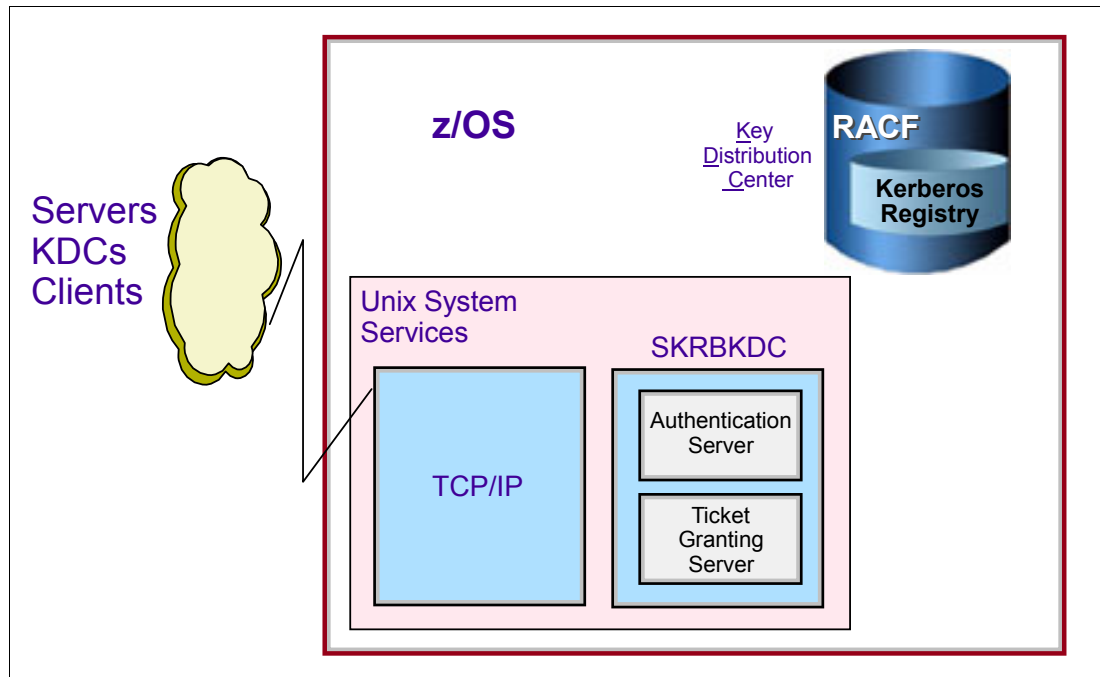


Figure 4-8 Some assumptions to Kerberos

### Some assumptions to Kerberos

The following assumptions apply to the Kerberos security environment:

- ▶ Denial-of-service (DoS) attacks are not addressed by Kerberos. There are places in these protocols where an intruder can prevent an application from participating in the proper authentication steps. Detection and solution of such attacks (some of which can appear to be “usual” failure modes for the system) is usually best left to human administrators and users.
- ▶ The secret key must be kept secret by each principal (each client and server). If an attacker steals a principal’s key, it can then masquerade as that principal or impersonate any server of the legitimate principal.
- ▶ Kerberos does not address password guessing attacks. If a poor password is chosen, an attacker might be able to mount an offline dictionary attack by repeatedly attempting to decrypt messages that are encrypted with a key derived from the user’s password.
- ▶ Kerberos assumes a loosely synchronized clock in the whole system. Workstations might be required to have a synchronization tool such as the time server provided.
- ▶ Principal identifiers should not be reused on a short-term basis. Instead, access control lists (ACLs) can be used to grant permissions to particular principals.

Figure 4-9 shows implementing Network Authentication Service.

## 4.9 Implementing Network Authentication Service

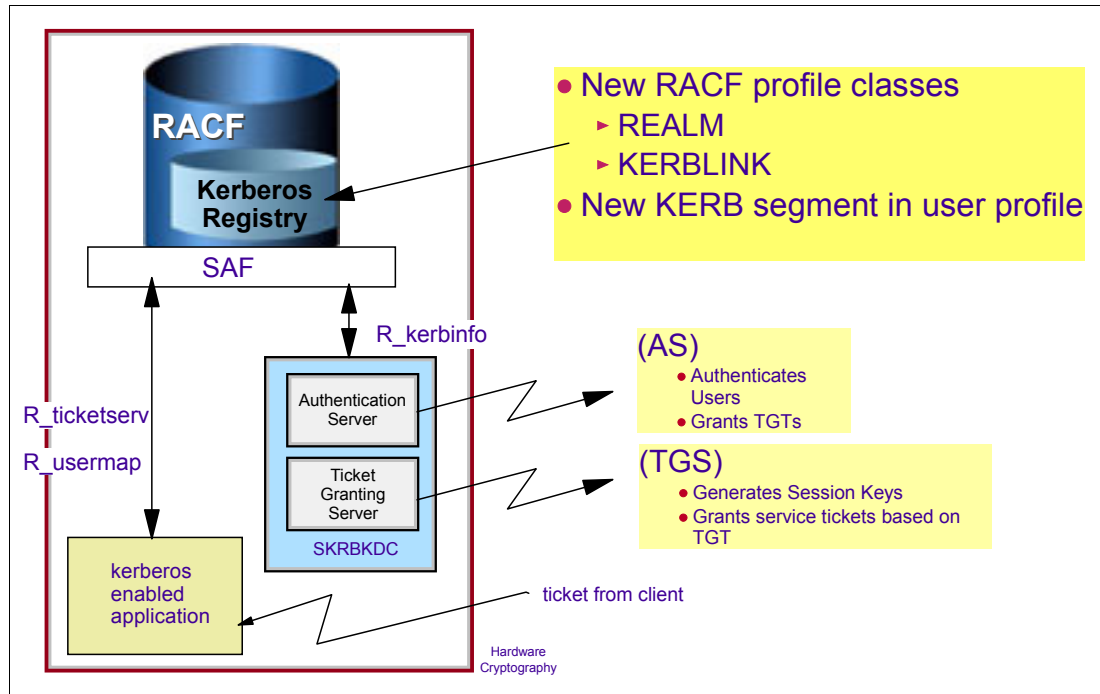


Figure 4-9 Implementing Network Authentication Service

### Implementing Network Authentication Service

The implementation of Network Authentication Service introduces a new UNIX daemon to provide the KDC services (authentication server and ticket-granting server). RACF has been enhanced to provide KDC registry functions to store principals and keys.

This section details the setup of this UNIX daemon and the required configuration files. It assumes that RACF is the incumbent ESM.

#### SKRBKDC daemon setup

The following steps describe the setup of the Network Authentication Service UNIX daemon called SKRBKDC:

1. Copy the SKRBKDC started task procedure (JCL) from the target library EUVF.SEUVFSAM to SYS1.PROCLIB or the procedure library you use in your installation for started tasks. Figure 4-10 on page 168 shows an example of the SKRBKDC started task procedure. By changing the PARMs value from **-kdc** to **-nokdc**, you can use it in a cut down mode and only provide application component trace or sysplex credential caches.

```

//*****
//*
//* Procedure for starting the Kerberos SKRBKDC started task
//*
//* Specify PARMS='-kdc' to enable the Kerberos KDC services.
//*
//* Specify PARMS='-nokdc' to disable the Kerberos KDC services.
//*
//*****
//SKRBKDC PROC REGSIZE=256M,OUTCLASS='A',PARMS='-kdc'
//*-----
//GO EXEC PGM=EUVFSKDC,REGION=&REGSIZE,TIME=1440,
// PARM=('ENVAR("LANG=En_US.IBM-1047"),TERM(DUMP) / &PARMS X
// 1>DD:STDOUT 2>DD:STDERR')
//STDOUT DD SYSOUT=&OUTCLASS,DCB=LRECL=250,
// FREE=END,SPIN=UNALLOC
//STDERR DD SYSOUT=&OUTCLASS,DCB=LRECL=250,
// FREE=END,SPIN=UNALLOC
//SYSOUT DD SYSOUT=&OUTCLASS,
// FREE=END,SPIN=UNALLOC
//CEEDUMP DD SYSOUT=&OUTCLASS,
// FREE=END,SPIN=UNALLOC

```

Figure 4-10 Example of the SKRBKDC started task procedure

2. Define a group for the started task user ID, with an OMVS segment with a GID value:

```

ADDGROUP SKRBGRP OWNER(STCGROUP) SUPGROUP(STCGROUP) +
DATA('GROUP FOR KERBEROS SKRBKDC User ID') OMVS(GID(20))

```

The owner that we specify in our examples is for our installation only. You might want to change this owner according to your installation standards.

To verify that the group is defined correctly, display the group with all the attributes as follows:

```

LISTGRP SKRBGRP OMVS

```

```

INFORMATION FOR GROUP SKRBGRP
SUPERIOR GROUP=STCGROUP OWNER=STCGROUP CREATED=13.218
INSTALLATION DATA=GROUP FOR KERBEROS SKRBKDC USER ID
NO MODEL DATA SET
TERMUACC
NO SUBGROUPS
NO USERS

OMVS INFORMATION
-----
GID= 0000000020

```

Figure 4-11 Display of Group SKRBGRP and its OMVS segment

3. Define a started task user ID with an OMVS segment with the following values:

- UID value: 0
- HOME (directory) value: /etc/skrb/home/kdc
- PROGRAM value: /bin/sh

**Attention:** Both the HOME and PROGRAM values are case-sensitive. You need to define them in lowercase.

An example definition is as follows:

```
ADDUSER SKRBKDC OW(SKRBGRP) DEFLTGRP(SKRGRP) NOPASSWORD +
NAME('KERBEROS User ID') OMVS(UID(0) HOME('/etc/skrb/home/kdc') +
PROG('/bin/sh'))
```

Use the RACF LISTUSER command to check that the user ID is correctly defined:

```
LISTUSER SKRBKDC OMVS
```

4. Activate the APPL class if not already active:

```
SETOPTS CLASSACT(APPL) RACLIST(APPL)
```

5. Define the SKRBKDC application to have a universal access of read. The alternate approach is for an UACC(NONE) and permit one or more groups read access. A user does require this access to perform the **kpasswd** Kerberos command to change their password.

```
RDEFINE APPL SKRBKDC UACC(READ)
```

6. Activate the PTKTDATA class if not already active:

```
SETOPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
```

7. Define PassTicket data to the SKRBKDC application. These PassTickets are used internally by the application when the user password is changed; it is not exposed externally. This secure signon key can be any valid DEK key as described in *z/OS Security Server RACF Security's Administrator's Guide*, SA23-2289.

```
RDEFINE PTKTDATA SKRBKDC UACC(NONE) SSIGNON(KEYMAKED(3734343237343131))
```

8. Refresh the APPL and PTKTDATA classes:

```
SETOPTS RACLISTC(APPL PTKTDATA) REFRESH
```

9. Define a profile for the SKRBKDC and SKBRWTR started tasks in the RACF STARTED class:

```
RDEFINE STARTED SKRBKDC.** OWNER(STCGROUP) +
STDATA(USER(SKRBKDC) GROUP(SKRBGRP))
RDEFINE STARTED SKBRWTR.** OWNER(STCGROUP) +
STDATA(USER(SKBRKDC) GROUP(SKBRGRP))
```

Check the new defined profile by listing it:

```
RLIST STARTED SKRBKDC.** STDATA
RLIST STARTED SKBRWTR.** STDATA
```

10. Refresh the STARTED class:

```
SETOPTS RACLISTC(STARTED) REFRESH
```

Setting up the Kerberos environment variable files is shown in Figure 4-12 on page 170.

## 4.10 Setting up the Kerberos environment variable files

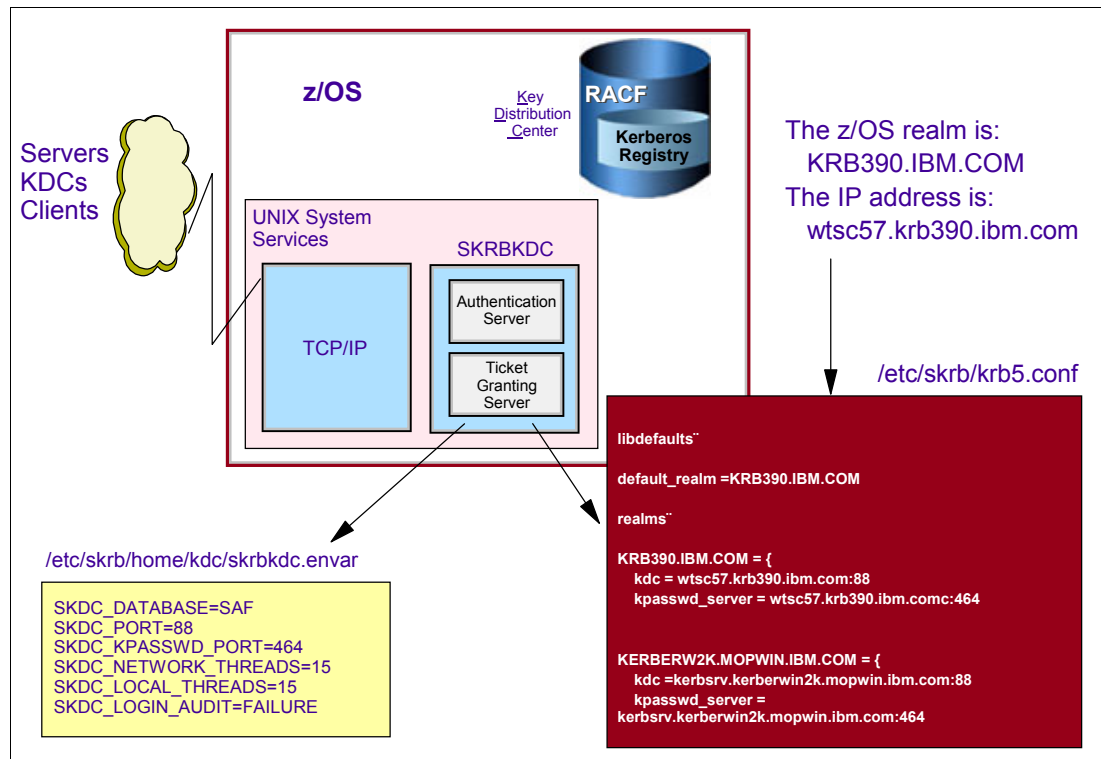


Figure 4-12 Setting up the Kerberos environment variable files

### Setting up the Kerberos environment variable files

You must customize the Kerberos environment variable files `/etc/skrb/krb5.conf` and `/etc/skrb/home/kdc/envar` for your environment. Samples of these configuration files are supplied in `/usr/lpp/skrb/examples`. Copy the samples to the locations indicated previously.

The `krb5.conf` file requires the following updates:

1. Update the `default_realm` parameter with your installation's Kerberos realm for the z/OS system. Our DNS name for our z/OS system is `WTSC57.KRB390.IBM.COM` and our Kerberos realm is `KRB390.IBM.COM`.
2. Update the `realms` parameter with your z/OS realms and any other so-called peer realms. We updated the `realms` parameter with our z/OS realm `KRB390.IBM.COM` and added the DNS name for the z/OS KDC and the z/OS `kpasswd_server`.
3. Update the `domain_realm` parameter to reflect the z/OS Realm in lowercase and uppercase.
4. `default_tkt_enctypes` is used for the encryption types for the session keys in the initial ticket granting tickets. In the supplied sample, they are listed from most preferred to least preferred. The selection should be highest, that is, support on the KDCs in the realm.
5. `default_tgs_enctypes` is used to encrypt session keys in service tickets.
6. The `use_dns_lookup` value is set not to use the DNS name server but use the `[realms]` and `[domain_realm]` sections of the configuration file.
7. The minimum ticket life is 15 seconds. The default ticket life is 10 hours, and the maximum ticket life is 24 hours.



Example 4-1 displays the configuration file, `/etc/skrb/krb5.conf`, and displays our changes to the configuration file.

*Example 4-1 Our changes to the `/etc/skrb/krb5.conf` file*

---

```
“libdefaults”
default_realm = KRB390.IBM.COM
kdc_req_checksum_type = rsa-md5
ap_req_checksum_type = rsa-md5
default_tgt_enctypes = des-cbc-crc,des-cbc-md5
default_tgs_enctypes = des-cbc-crc,des-cbc-md5
kdc_default_options = 0x40000010
use_dns_lookup = 0
“realms”
KRB390.IBM.COM = {
    kdc = wtsc57.krb390.ibm.com:88
    kpasswd_server = wtsc57.krb390.ibm.com:464
}
KRB2000.IBM.COM = {
    kdc = pauldeg.krb2000.ibm.com:88
    kpasswd_server = pauldeg.itso.ibm.com:464
}
“domain_realm”
.krb2000.ibm.com = KRB2000.IBM.COM
.krb390.ibm.com = KRB390.IBM.COM
```

---

The next step is to configure the environment variable file `/etc/skrb/home/kdc/envar` with the required changes for your environment.

The defaults in this file are usually fine, except perhaps the time zone and the required logging that you want to perform for the Kerberos server (SKRKBKDC). The `SKDC_DATABASE` value of `SAF` implies that `RACF` (and possibly any other `ESM`) is to provide a registry function. This is what IBM recommends unless it is necessary to share the Kerberos registry with other KDC instances on other operating systems.

The environment variable `SKDC_TKT_ENCTYPES` processes the list of encryption types from left to right until requirements are met. KDC attempts to use the same encryption algorithm for the service ticket as was used for the ticket granting ticket.

Example 4-2 shows an example of the environment variable definitions for the Kerberos server.

*Example 4-2 Example of the environment variable definitions*

---

```
General server options
SKDC_DATABASE=SAF
SKDC_PORT=88
SKDC_KPASSWD_PORT=464
SKDC_KADMIN_PORT=749
SKDC_NETWORK_THREADS=15
SKDC_LOCAL_THREADS=15
SKDC_LOGIN_AUDIT=FAILURE
SKDC_CONSOLE_LEVEL=E
# Enable DES encryption types (AES, DES3 and DESD are disabled)
SKDC_TKT_ENCTYPES=des-cbc-md5,des-cbc-md4,des-cbc-crc
#
# System configuration options
#
LANG=En_US.IBM-1047
```

```
TZ=EST5EDT
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/En_US.IBM-1047/%N

#
# Message/debug options
#
_EUV_SVC_MSG_LOGGING=STDOUT_LOGGING
_EUV_SVC_MSG_IDENTITY=SKRBKDC
_EUV_SVC_MSG_FACILITY=AUTH
_EUV_SVC_DBG_MSG_LOGGING=1
_EUV_SVC_DBG=KRB_KDC.8,KRB_KDB.8
_EUV_EXC_ABEND_DUMP=0
```

---

For these configuration files in /etc/skrb:

- ▶ All files are only to be updated by your administrator (replace xxx with user ID).  
chown -R xxx /etc/skrb/
- ▶ All configuration files (exceptions to follow) only need to be readable by the KDC.  
chmod -R 600 /etc/skrb
- ▶ The two directories need to have execute permissions so the administrator can change into them.  
chmod 700 /etc/skrb/home /etc/skrb/home/kdc
- ▶ krb5.conf (and the parent directory) needs to be readable by everyone.  
chmod 644 /etc/skrb/krb5.conf  
chmod 755 /etc/skrb
- ▶ Any keytabs in /etc/skrb need to allow the application server (not client) read access.

This completes the setup for the Kerberos server, but before you start the Kerberos server, some additional RACF definitions are required.

Setting up hierarchical file system (HFS) for Kerberos cache files is shown in Figure 4-13 on page 173.

## 4.11 Setting up HFS for Kerberos cache files

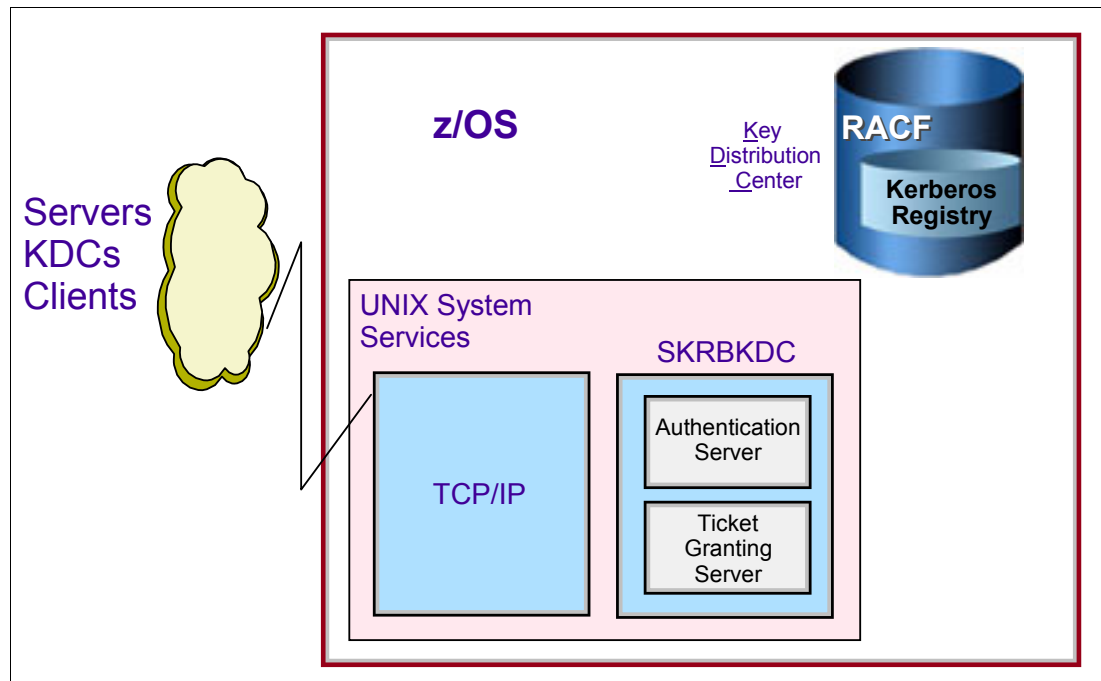


Figure 4-13 Setting up HFS for Kerberos cache files

### Setting up HFS for Kerberos cache files

The Kerberos runtime stores network credentials in so-called cache files. These are stored in an HFS directory called `/var/skrb/creds`. This directory structure does not exist by default and requires setup. Also, these files need to be erased periodically. There are several ways to erase the files:

- ▶ Use a temporary file system mounted at `/var/skrb/creds`. This results in all the credentials cache files being deleted each time the system is restarted.
- ▶ Erase all of the files in `/var/skrb/creds` when the `/etc/rc` initialization script is run. This results in all of the credentials cache files being deleted each time the system is restarted.
- ▶ Set up a cron job to run the `kdestroy` command with the `-e` option. This results in the deletion of only expired credentials cache files. This is the preferred method for managing the credentials cache files. The cron job should run with UID 0 so that it can delete the cache files.

The `/var/skrb/creds` directory permission bits should be set to 777 using the `chmod` command:

```
chmod 777 /var/skrb/creds
```

Figure 4-14 on page 174 shows Kerberos integrated with RACF.

## 4.12 Kerberos integrated with RACF

- RACF must be setup as a local RRSF node
- Definition of RACF profiles
  - Definition of the local Kerberos realm& foreign realms
    - **REALM class**
  - Local Kerberos principals (users)
    - **KERB segment in user profile**
    - **KERBLINK class profiles**
  - Definition of foreign Kerberos principals with a local identity
    - **KERBLINK class profiles**

Figure 4-14 Kerberos integrated with RACF

### Kerberos integrated with RACF

The Kerberos security server supports two registry database types: SAF (for example RACF) and NDBM (Kerberos principals stored in a UNIX System Services database using HFS or zFS). IBM recommends that you use the SAF registry unless it is necessary to share the Kerberos registry with one or more KDC instances running on another operating system.

If SAF is selected, RACF provides the functions to customize and access data for use with Kerberos. Then, the z/OS Network Authentication Service server maintains registry of principal and global information, which is stored using RACF through User and General Resource Profiles.

You can administer the Network Authentication Service server through the RACF panels and commands and obtain this information through an SAF callable service. Kerberos application servers can use SAF callable services to parse Kerberos tickets to obtain principal names, and to map from principal to RACF user and vice versa.

Local Kerberos principals are defined as RACF users with a KERB segment. The information about the local and foreign realms are defined in the RACF class REALM in specific profiles. The profiles contain:

- ▶ Local realm information, the name, key, and ticket lifetime (MIN, MAX, and DEFAULT in seconds).
- ▶ Foreign realm trust relationships. These are defined in pairs, which also include a key. RACF maps foreign Kerberos principals using the KERBLINK class profiles.

The Kerberos principal's password and the RACF user password are integrated. The Kerberos password is subject to RACF SETROPTS rules and installation-defined rules.

Principals must keep their secret keys secret. If an intruder steals a principal's key, it can then masquerade as that principal or impersonate any server to the legitimate principal.

RACF Remote Sharing Facility (RRSF) must be defined in local mode to generate the corresponding Kerberos secret key whenever the user changes their password. Kerberos uses RRSF services to make sure this happens.

Some RRSF RACF functions require a previously established user ID association. A user ID association is an association between two or more user IDs on the same or different RRSF nodes.

There are two types of user ID associations:

- ▶ A *peer association* allows either of the associated user IDs to direct commands to the other and allows password synchronization.
- ▶ In a *managed association*, one of the user IDs is designated as the managing ID, and the other is designated as the managed ID. The managed ID cannot direct commands to the managing ID. There is no password synchronization in a managed association.

To use the password synchronization and command direction functions, you need to activate and define profiles in to the RRSFDATA class.

## Defining RRSF in local mode

To define RRSF in local mode, follow these steps:

1. Activate the RACF RRSFDATA class if it is not activated already. The RRSFDATA class needs to be RACLISTed and activated for generic command processing:

```
SETROPTS CLASSACT(RRSFDATA)
SETROPTS GENERIC(RRSFDATA)
SETROPTS GENCMD(RRSFDATA)
SETROPTS RACLIST(RRSFDATA)
```

2. Define a new member IRROPT01 in SYS1.PARMLIB and include the **TARGET** command as shown in Example 4-3 to configure the RRSF in local mode. This to define your local system as the local RRSF node. It is required to allow the generation of keys for local principals who change their own passwords.

*Example 4-3 Define a new member*

---

```
TARGET -
NODE(SC57) -
DESCRIPTION('WS57TS SYSTEM') -
PREFIX(SYS1.RACF) -
OPERATIVE LOCAL -
WORKSPACE(VOLUME(PDGTS1))
```

---

3. Modify the RACF procedure in SYS1.PROCLIB to process the updated RACF parameter library by adding PARM='OPT=01' to the EXEC statement. Add the RACFPARM ddname to point to SYS1.PARMLIB to identify the library that contains the RRSF parameters. Example 4-4 shows the RACF procedure in SYS1.PROCLIB.

*Example 4-4 The RACF procedure in SYS1.PROCLIB*

---

```
//RACF PROC
//RACF EXEC PGM=IRRSSM00,REGION=0M,PARM='OPT=01'
//RACFPARM DD DSN=SYS1.PARMLIB,DISP=SHR
```

---

4. For these changes to take effect, refresh by taking the following steps:
  - a. Refresh the RACF subsystem by stopping and restarting the RACF started task using the locally defined RACF subsystem prefix. This would be used only when considering configuring a new main system in a multisystem node due to its complex nature.

Issue the **MVS START** command, specifying RACF as the procedure name:

```
S RACF,SUB=MSTR
```

- b. Normally you would issue the command on the affected z/OS system:

```
SET INCLUDE(xx) where XX is the suffix of the IRROPTxx member
```

5. You can check the status of the RRSF environment using the **TARGET LIST** command, using the locally defined RACF subsystem prefix:

```
#TARGET LIST
```

6. You can also check the RACF subsystem using the **SET LIST** command:

```
#SET LIST
```

## RACF setup for Kerberos realms

To set up RACF for Kerberos realms, follow these steps:

1. Before you define the local REALM, activate (CLASSACT) and RACLIST the REALM class:

```
SETROPTS CLASSACT(REALM)
SETROPTS RACLIST(REALM)
```

2. Activate, if not already active, and RACLIST the PTKTDATA class as follows:

```
SETROPTS CLASSACT(PTKTDATA)
SETROPTS GENERIC(PTKTDATA)
SETROPTS GENCMD(PTKTDATA)
SETROPTS RACLIST(PTKTDATA)
```

**Important:** No profile is needed in the PTKTDATA class. The Kerberos server (SKRKBKDC) generates a temporary PassTicket under the covers to change a principal's password when the **kpasswd** command is issued.

3. A user must have access to the SKRKBKDC application in order to use the **kpasswd** command to change their password. By using the **RACF RDEFINE** command, you can define the SKRKBKDC application to the RACF APPL class:

```
RDEFINE APPL SKRKBKDC OWNER(SYS1) UACC(READ) +
DATA('KERBEROS APPLID')
```

**Tip:** Alternately, you can set the universal access to NONE and explicitly authorize individual groups or users to the SKRKBKDC application.

4. Define your local realm to the REALM class, using the **RACF RDEFINE** command to define the KERBDFLT profile reflecting the default REALM and policy:

```
RDEFINE REALM KERBDFLT KERB(KERBNAME(KRB390.IBM.COM) PASSWORD(password) +
MINTKTLFE(15) DEFTKTLFE(36000) MAXTKTLFE(86400)
```

**Attention:** Our z/OS environment has a domain name of WTSC57.KRB390.IBM.COM and a REALM name of KRB390.IBM.COM

Use the **RACF RLIST** command to display the KERBDFLT profile in the REALM class:

```
RLIST REALM KERBDFLT KERB
```

```
CLASS      NAME
-----
REALM      KERBDFLT
```

```
KERB INFORMATION
-----
KERBNAME=  KRB390.IBM.COM
MINTKTLFE= 0000000015
```

```
MAXKTLFE= 000086400
DEFTKTLFE= 000036000
KEY VERSION= 001
KEY ENCRYPTION TYPE= DES DES3 DESD AES128 AES256
CHECK ADDRESSES= NO
```

5. Define any foreign REALMs to the RACF REALM class.

Your local Network Authentication and Privacy Service (Kerberos) server can trust authentications completed by other servers, and can be trusted by other servers, by participating in trust relationships.

To participate in trust relationships, you must define each server as a foreign realm. Then, you can allow users who are authenticated in foreign realms (foreign principals) to access protected resources on your local z/OS system by mapping one or more RACF user IDs to foreign principal names. You do not need to provide foreign principals with the ability to log on to your local z/OS system. You can simply provide mapping to one or more local user IDs so they can gain access privileges for local resources that are under the control of an z/OS application server, such as DB2.

In our example, we defined a Windows 2000 Realm to the RACF REALM class so that we can use it later for testing the Kerberos integration between Windows 2000 and the z/OS Network Authentication and Privacy Services using DB2.

The Windows 2000 domain is called pauldeg.krb2000.ibm.com and the REALM is called KRB2000.IBM.COM. We defined the following profiles to set up the trust relationship between the z/OS REALM and the Windows 2000 REALM:

```
RDEFINE REALM /.../KRB390.IBM.COM/krbtgt/KRB2000.IBM.COM + KERB(PASSWORD(xx))
RDEFINE REALM /.../KRB2000.IBM.COM/krbtgt/KRB390.IBM.COM + KERB(PASSWORD(xx))
```

**Attention:** You need the password that is defined here later when you define the same trust relationship on the Windows 2000 domain. This password is not associated with any user ID and is not constrained to any SETROPTS rules for passwords.

6. Define Kerberos port 88 for the KDC and port 464 for the password server to your TCP/IP profile to reflect the use of these ports, as shown in Example 4-5.

*Example 4-5 Define Kerberos port 88 for the KDC and port 464 for the password server*

---

```
88 TCP OMVS SAF KERB88 ; Kerberos Server
464 TCP OMVS SAF KERB464 ; Kerberos Server
```

---

7. Depending on your installation, you might or might not have started with the protection of TCP/IP ports using the RACF SERVAUTH class. Accordingly, you should authorize the SKRBKDC Started Task User ID to port 88 and 464, using the following commands:

```
PERMIT EZB.PORTACCESS.SC57.ITCPIP.KERB88 CLASS(SERVAUTH) +
ID(SKRBKDC) ACCESS(READ)
PERMIT EZB.PORTACCESS.SC57.ITCPIP.KERB464 CLASS(SERVAUTH) +
ID(SKRBKDC) ACCESS(READ)
```

The SKRBKDC started task also requires access to the TCP/IP stack itself, using a new profile in the RACF SERVAUTH class:

```
PERMIT EZB.STACKACCESS.SC57.ITCPIP CLASS(SERVAUTH) ID(SKRBKDC) + ACCESS(READ)
```

As an alternative if SERVAUTH is not used to protect TCP/IP ports, it should be considered that the Communications Server profile member be updated to reserve the ports for KDC (usually 88), KPASSWD (usually 464), and KADMIN (usually 749). While the profile is used to reserve ports, entries in the Communications Server /etc/services

can assign ports to Kerberos services. It should be checked to see the port assignments in it for Kerberos match your installation, the file is called /etc/services.

8. At this point, we need to refresh profiles in storage:

```
SETROPTS CLASS(REALM APPL) REFRESH
```

9. You are now ready to start your Kerberos server SKRBKDC. You receive the informational messages shown in Example 4-6. Ensure that the load library is APF-authorized.

*Example 4-6 Start Kerberos server: Informational messages*

---

```
S SKRBKDC
$HASP100 SKRBKDC ON STCINRDR
IEF695I START SKRBKDC WITH JOBNAME SKRBKDC IS ASSIGNED TO USER
SKRBKDC , GROUP SKRBGRP
$HASP373 SKRBKDC STARTED EUVF04001I Security server version 2.10, Service
level 0W45102.
EUVF04002I Security runtime version 2.10, Service level 0W45102.
EUVF04018I Security server initialization complete.
```

---

Define Kerberos local principals is shown in Figure 4-15.

## 4.13 Define Kerberos local principals

▪ Define local principals

- ALTUSER user1 KERB(KERBNAME(KerbUSER1)) PASSWORD(usrp) NOEXPIRED

user profile SUPUSER, to be used as the DB2 server userid  
kerbname = DBPRINCIPAL  
password= password3

The service ticket that we receive will address the DB2 server with principal name 'DBPRINCIPAL'

Figure 4-15 Kerberos principals: Local principals

### Define Kerberos principals

This section describes how to define Kerberos principals. We distinguish between two types of principals:

- ▶ A *local* principal is a Kerberos user defined to the local REALM.
- ▶ A *foreign* principal is a Kerberos user from another Kerberos REALM.



## Local principals

You define local principals as RACF users using the **ADDUSER** and **ALTUSER** commands with the new **KERB** option. This creates a KERB segment for the user ID. Each local principal must have a RACF password. Therefore, do not use the **NOPASSWORD** option when defining local principals. You can specify the following information for your local principals:

- ▶ **KERBNAME**: Local principal name.
- ▶ **MINTKTLFE**: Minimum ticket lifetime for the local realm.
- ▶ **MAXTKTLFE**: Maximum ticket lifetime for the local principal.
- ▶ **CHECKADDRS**: KDC to check addresses in tickets as part of ticket validation processing. This should be disabled (default) if your requests pass through routers or firewalls using Network Address Translation (NAT).
- ▶ **ENCRYPT**: Specifies which keys the local principal is allowed to use. The current supported key types are DES, DES3, DESD, AES128, AES256.
- ▶ **PASSWORD**: The password for the realm.

**Important:** Upper and lowercase letters are accepted and maintained in the case in which they are entered.

Example 4-7 shows an example of defining a Kerberos principal.

*Example 4-7 Defining a Kerberos principal*

---

```
ALTUSER GRAAFF KERB(KERBNAME('Paul de Graaff'))
```

---

**Restriction:** You can define the local principal name that you specify only once. If you try to define it to two RACF user IDs, you receive the following error message:

```
IRR52165I The value for the KERB segment KERBNAME operand must be unique.  
Command processing ends.
```

## Generating keys for local principals

Each local principal must have a key registered with the local Network Authentication and Privacy Service (Kerberos) server in order to be recognized as a local principal. The user's definition as a local principal is not complete until the key is generated. The key is generated from the principal's RACF user password at the time of the user's password change. If you want a key to be generated, be sure to use a password change facility that will not result in an expired password that the user must change at next logon. For example, you can use the **NOEXPIRED** keyword of the **ALTUSER** command.

A local principal's key is revoked whenever the user's RACF user ID is revoked or the RACF password is considered expired. If the user's key is revoked, the server rejects ticket requests from this user.

You can change a user's password so that a key can be generated using the **ALTUSER** command with the **NOEXPIRED** option, for example:

```
ALTUSER GRAAFF PASSWORD(new1pw) NOEXPIRED
```

**Attention:**

- ▶ Do not use the **NOPASSWORD** option on the **ALTUSER** command.
- ▶ You must specify a password value so that a key can be generated. All characters of the password are folded to uppercase.

Users can change their own passwords by completing their own definitions as local principals by using any standard RACF password-change facility, such as:

- ▶ TSO PASSWORD command (without the ID option)
- ▶ TSO logon
- ▶ CICS signon

**Important:** The RACF address space must be started for the password change to complete and the key to be generated.

Password change requests from applications that encrypt the password before calling RACF do not result in usable keys.

### Automatic local principal name mapping

For each local principal that you define on your system using the KERB keyword of the **ADDUSER** and **ALTUSER** commands, RACF creates a mapping profile in the KERBLINK class automatically. When you issue the **ALTUSER** command with the NOKERB keyword or issue a **DELUSER** for a user with a KERB segment, RACF deletes the KERBLINK profile automatically.

The KERBLINK profile maps the local principal name to the user's RACF user ID. The name of the KERBLINK profile for a local principal is the principal name specified as the KERBNAME value with the **ADDUSER** or **ALTUSER** command. We show the KERBLINK profile for user ID graaff in Example 4-8 as it was defined in Example 4-7 on page 179.

*Example 4-8 The KERBLINK profile for user ID graaff*

---

```
sr mask(P) class(kerblink)
PaulçdeçGraaff
```

---

You can see in the profile PaulçdeçGraaff that blanks are indeed replaced by the ç character. If you list the profile, you notice a little quirk in the RACF command processing where it does not accept mixed-case profile names, as shown in Example 4-9.

*Example 4-9 Example with mixed-case profile names*

---

```
r1 KERBLINK PaulçdeçGraaff
ICH13003I PAULçDEçGRAAFF NOT FOUND
```

---

If you do an **RLIST \***, you see the output shown in Example 4-10.

*Example 4-10 Output of RLIST \**

---

```
RLIST *
CLASS NAME
-----
KERBLINK PaulçdeçGraaff
LEVEL OWNER UNIVERSAL ACCESS YOUR ACCESS WARNING
-----
OO GRAAFF NONE NONE NO
INSTALLATION DATA
-----
NONE
APPLICATION DATA
-----
GRAAFF
```

---

Example 4-10 on page 180 shows that the local principal, Paul de Graaff, maps back to RACF user ID GRAAFF.

### Considerations for local principal names

The name of the KERBLINK profile contains the local principal name that is mapped. Local principal names can contain embedded blanks and lowercase characters.

Blanks are not permitted as a part of a RACF profile name. Therefore, when building the KERBLINK profile name, as a result of specifying KERBNAME with the **ADDUSER** or **ALTUSER** command, RACF command processing replaces each blank with the X'4A' character (which often resolves to the ¢ symbol), as shown in the output from the RLIST KERBLINK \* command shown in Example 4-8 on page 180, and in the output from the RACF database unload utility (IRRDBU00).

**Restriction:** RACF command processing also prevents the X'4A' character from being specified as part of the actual local principal name.

Define Kerberos foreign principals is shown in Figure 4-16.

## 4.14 Define Kerberos foreign principals

- Define foreign principals
  - RDEFINE KERBLINK /.../foreign\_realm/foreign\_principal APPLDATA('racf\_user')
  - maps single principal to a RACF user
  - RDEFINE KERBLINK /.../foreign\_realm/ APPLDATA('racf\_user')
  - Maps all principals for a single realm to a RACF userid

```
KERBLINK /.../KERBERW2K.MOPWIN.IBM.COM/LAMBDA APPLDATA('CLIENT1')
```

RACF will map the Windows DB2 user Kerberos principal name LAMBDA to RACF userid CLIENT1

Figure 4-16 Kerberos principals: Foreign principals

### Kerberos foreign principals

You map foreign principal names to RACF user IDs on your local z/OS system by defining general resource profiles in the KERBLINK class. You can map each principal in a foreign realm to its own user ID on your local z/OS system, or you can map all principals in a foreign realm to the same user ID on your system.

RACF user IDs that map to foreign principals do not need KERB segments. These user IDs are intended to be used only to provide local z/OS identities to associate with access privileges for local resources that are under the control of an z/OS application server, such as DB2.

Each mapping profile in the KERBLINK class is defined and modified using the RDEFINE and RALTER commands. The name of the KERBLINK profile for a foreign principal contains the principal name, fully qualified with the name of the foreign realm.

The profile name uses the following format:

```
.../foreign_realm / [foreign-principal_name ]
```

If you want to map a unique RACF user ID to each foreign principal, you must specify the foreign realm name and the foreign principal name. If you want to map the same RACF user ID to every foreign principal in the foreign realm, you need only specify the foreign realm name. In each case, you specify the local user ID using the APPLDATA keyword of the **RDEFINE** or **RALTER** command.

### Example of mapping foreign principal names

In the following example, the users PAUL and VAL have their foreign principal names mapped with individual user IDs on the local z/OS system. All other foreign principals presenting tickets from the KERB2000.IBM.COM REALM are mapped to the KRB2000 user ID on the local z/OS system.

```
RDEFINE KERBLINK /.../KERB2000.IBM.COM/PAUL APPLDATA('GRAAFF')
RDEFINE KERBLINK /.../KERB2000.IBM.COM/VAL APPLDATA('VALERIA')
RDEFINE KERBLINK /.../KERB2000.IBM.COM/ APPLDATA('KERB2000')
```

**Attention:** All characters of the foreign realm name and the foreign principal name are folded to uppercase.

Figure 4-17 shows the Kerberos user commands.

## 4.15 Kerberos user commands

- kinit
- klist
- kdestroy
- keytab
- ksetup
- kpasswd
- kvno
- kadmin

Figure 4-17 Kerberos user commands

### Description of the Kerberos commands

The following commands are supplied:

- ▶ **kinit**: Obtains or renews a Kerberos ticket-granting ticket. The KDC options specified in the Kerberos configuration file are used if no ticket options are specified on the kinit command.
- ▶ **klist**: Displays the contents of a Kerberos credentials cache or key table.

- ▶ **kdestroy**: Destroys a Kerberos credentials cache file. To delete a credentials cache, the user must be the owner of the file or must be a root (uid) user.
- ▶ **keytab**: Used to add or delete a key from a key table or to display the entries in a key table.
- ▶ **kpasswd**: Changes the password for a Kerberos principal using the password change service.
- ▶ **kvno**: Displays the current key version number for a principal.
- ▶ **kadmin**: Is used to manage entries in the Kerberos database. It prompts you to enter one or more subcommands.
- ▶ **utility commands**: We have kpropd, which is used to “catch” a stand-alone database propagation. We also have kdb5\_ndbm, which is the Kerberos NDBM database maintenance utility.

To use these commands, you must update your PATH statement in your .profile with the full path name of the directory (/usr/lpp/skrb/bin) containing the Kerberos commands. It also requires updates to the NLSPATH statement to reflect the Kerberos message catalog. Example 4-11 displays the required changes to your .profile.

*Example 4-11 Required changes to .profile*

---

```
# =====
# Start of Kerberos section
# =====
echo "--> Start of Kerberos Additions"
export PATH=$PATH:/usr/lpp/skrb/bin
echo "PATH:" $PATH
#
export NLSPATH=$NLSPATH:/usr/lpp/skrb/lib/nls/msg/%L/%N
echo "NLSPATH:" $NLSPATH
#
```

---

## Kerberos command examples

The section that follows lists some examples of Kerberos commands.

### The kinit command

The **kinit** command obtains or renews the Kerberos ticket-granting ticket. The KDC options specified by `kdc_default_options` in the Kerberos configuration file are used if no ticket options are specified on the **kinit** command. The **kinit** command includes several keywords, but we show only the following examples here:

- ▶ **kinit -s**: Obtains a ticket-granting ticket using the current signed-on RACF user ID
- ▶ **kinit**: Obtains a ticket-granting ticket and the principal name is obtained from the credentials cache (if present)
- ▶ **kinit -k**: Obtains a ticket-granting ticket using a key table to obtain the principal information

#### ***kinit -s example***

To obtain a ticket-granting ticket for a Kerberos principal, you can either use RACF services to obtain the principal associated or use a so-called key table. The **kinit -s** command obtains a ticket-granting ticket for the current signed-on RACF user ID, as shown in Example 4-12 on page 184.

*Example 4-12 Example of kinit -s to obtain ticket-granting ticket*

---

```
GRAAFF @ SC57:/u/graaff/kerberos>kinit -s
EUVF06014E Unable to obtain initial credentials.
          Status 0x96c73a2d - Service key is not available.
```

---

When we issue the **kinit -s** command for the current signed-on RACF user ID GRAAFF, we receive an error that the service key is not available. When we define the local principal for user ID GRAAFF, a key is not generated for the local principal. When we issue the LU GRAAFF KERB command, no key is generated, as shown in Example 4-13.

*Example 4-13 Issuing the LU GRAAFF KERB command*

---

```
LU GRAAFF KERB NORACF
USER=GRAAFF
KERB INFORMATION
-----
KERBNAME= Paul de Graaff
```

---

Keys get generated only when a RACF password change occurs. So after we change the password for the RACF user ID GRAAFF, we receive a key generated, as shown in Example 4-14.

*Example 4-14 Generated key for user ID GRAAFF*

---

```
LU GRAAFF KERB NORACF
USER=GRAAFF
KERB INFORMATION
-----
KERBNAME= Paul de Graaff
KEY VERSION= 001
```

---

We can now try again to get a ticket-granting ticket issued, using the **kinit -s** command, as shown in Example 4-15.

*Example 4-15 Using the kinit -s command to get a ticket-granting ticket*

---

```
GRAAFF @ SC57:/u/graaff/kerberos>kinit -s
GRAAFF @ SC57:/u/graaff/kerberos>klist
Ticket cache: FILE:/var/skrb/creds/krbcred_a9b31900
Default principal: Paul de Graaff@KRB390.IBM.COM
Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
Valid 2001/02/26-23:48:16 to 2001/02/27-09:48:16
```

---

***kinit with no keywords example***

Next, we tested the **kinit** command without specifying any keywords. The **kinit** command obtains the principal name from the credentials cache. If no credential cache exists, the command fails, as shown in Example 4-16.

*Example 4-16 Failure of the kinit command without specifying keywords*

---

```
GRAAFF @ SC57:/u/graaff>kinit
EUVF06010E Principal name must be specified.
```

---

Example 4-17 shows the interaction with the user when issuing the `kinit` command and a credential cache does exist. You are prompted for a password associated with the local principal. If you are using RACF instead of a key table for storage of local principals, this is your RACF password that is associated with your RACF user ID.

*Example 4-17 Using the `kinit` command when a credential cache does exist*

```
GRAAFF @ SC57:/u/graaft>kinit
EUVF06017R Enter password:
GRAAFF @ SC57:/u/graaft>
```

**Important:** You must enter the password here in uppercase letters. RACF accepts only uppercase passwords.

### ***kinit -k example***

We then tested the use of a key table with the `kinit` command rather than using RACF. For this example, we assume a principal is defined called `paul@KRB390.IBM.COM`. Example 4-18 shows using a key table with the `kinit` command, by using the `-k` keyword.

*Example 4-18 Using the `kinit -k` command*

```
GRAAFF @ SC57:/u/graaft>kinit -k paul@KRB390.IBM.COM
GRAAFF @ SC57:/u/graaft>klist
    Ticket cache: FILE:/var/skrb/creds/krbcred_b210de60
    Default principal: paul@KRB390.IBM.COM
Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
    Valid 2001/06/08-13:39:50 to 2001/06/08-23:39:50
GRAAFF @ SC57:/u/graaft>
```

**Note:** You must enter the password in uppercase letters.

When using the `-k` keyword, you do not need to specify the name and location of the key table if you want to use the default key table. The default key table name is obtained from the `default_keytab_name` configuration file (`krb5.conf`) entry. The default name is `/etc/skrb/krb5.keytab`.

**Tip:** You can also change the default key table name using the environment variable `KRB5_KTNAME`.

### **The `klist` command**

Using the `klist` command displays the contents of a Kerberos credentials cache or key table. We show the following examples of using the `klist` command:

<code>klist</code>	Lists the tickets in the credentials cache (the default).
<code>klist -e</code>	Displays the encryption type for the session key and the ticket.
<code>klist -f</code>	Displays the ticket flags.
<code>klist -k</code>	Displays the entries in the keytable.
<code>klist -k -K</code>	Displays the encryption key value for each key table entry.

### ***klist example***

When you issue the `klist` command without any keywords, it is as though you had issued a `klist -c` command. Example 4-19 on page 186 shows the output of a `klist` command after a ticket-granting ticket is obtained.

*Example 4-19 The klist command example*

---

```
GRAAFF @ SC57:/u/graaff/kerberos>kinit -s
GRAAFF @ SC57:/u/graaff/kerberos>klist
  Ticket cache: FILE:/var/skrb/creds/krbcred_a9b31900
  Default principal: Paul de Graaff@KRB390.IBM.COM
Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
  Valid 2001/02/26-23:48:16 to 2001/02/27-09:48:16
```

---

***klist -e example***

The **klist -e** command displays the encryption type for the session key and the ticket, as shown in Example 4-20.

*Example 4-20 The klist -e command example*

---

```
GRAAFF @ SC57:/u/graaff>klist -e
  Ticket cache: FILE:/var/skrb/creds/krbcred_b210de60
  Default principal: paul@KRB390.IBM.COM
Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
  Valid 2001/06/08-13:39:50 to 2001/06/08-23:39:50
  Encryption type: DES_CBC_CRC
```

---

***klist -f example***

The **klist -f** command displays the ticket flags, as shown in Example 4-21.

*Example 4-21 The klist -f command example*

---

```
GRAAFF @ SC57:/u/graaff>klist -f
  Ticket cache: FILE:/var/skrb/creds/krbcred_b210de60
  Default principal: paul@KRB390.IBM.COM
Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
  Valid 2001/06/08-13:39:50 to 2001/06/08-23:39:50
  Flags: FIA
```

---

In this example, the flags indicate:

F	Forwardable ticket
I	Initial ticket
A	Preauthentication used

**Important:** The **-f** option is valid only when listing a credentials cache.

***klist -k example***

The **klist -k** command lists the entries in a key table, as shown in Example 4-22.

*Example 4-22 The klist -k example*

---

```
GRAAFF @ SC57:/u/graaff>klist -k
  Key table: /etc/skrb/krb5.keytab
Principal: paul@KRB390.IBM.COM
  Key version: 1
```

---

***klist -k -K example***

The **klist -k -K** command lists the entries in a key table and displays the encryption key value for each key table entry, as shown in Example 4-23 on page 187.



#### Example 4-23 The `klist -k -K` example

---

```
GRAAFF @ SC57:/u/graaff>klist -k -K
Key table: /etc/skrb/krb5.keytab
Principal: paul@KRB390.IBM.COM
Key version: 1
Key: f1bc4fa49e4975ad
```

---

### The `kdestroy` command

The `kdestroy` command deletes a Kerberos credentials cache file.

The `-e` option causes the `kdestroy` command to check all of the credentials cache files in the default cache directory (`/etc/skrb/var/creds`). Any file that contains only expired tickets that have expired for the time delta value are deleted. The time delta is expressed as *nwndnhmms*, where:

<i>n</i>	Represents a number
<i>w</i>	Indicates weeks
<i>d</i>	Is days
<i>h</i>	Is hours
<i>m</i>	Is minutes
<i>s</i>	Indicates seconds

The components must be specified in this order, but any component can be omitted (for example, `4h5m` represents 4 hours and 5 minutes, and `1w2h` represents 1 week and 2 hours). If only a number is specified, the default is hours.

**Important:** To delete a credentials cache, the user must be the owner of the file or must be a root user (uid 0).

Example 4-24 shows an example of the `kdestroy` command that deletes the credentials cache of principal Paul de Graaff.

#### Example 4-24 Example of the `kdestroy` command

---

```
GRAAFF @ SC57:/u/graaff>kinit -s
GRAAFF @ SC57:/u/graaff>klist
Ticket cache: FILE:/var/skrb/creds/krbcred_b2118de0
Default principal: Paul de Graaff@KRB390.IBM.COM
Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
Valid 2001/06/08-14:26:38 to 2001/06/09-00:26:38
GRAAFF @ SC57:/u/graaff>kdestroy
EUVF06034I Credentials cache FILE:/var/skrb/creds/krbcred_b2118de0
destroyed.
```

---

### The `keytab` command

The `keytab` command manages a key table. A key table can be used to define either local or foreign principals. Key tables are traditionally used in UNIX based environments. Support for key tables here provides compatibility with these environments.

To define the local principal `graaff` to the default key table, issue the `keytab` command, as shown in Example 4-25 on page 188.

*Example 4-25 The keytab command example*

---

```
GRAAFF @ SC57:/u/graaff>keytab add paul -p paul
GRAAFF @ SC57:/u/graaff>keytab list paul
  Key table: /etc/skrb/krb5.keytab
Principal: paul@KRB390.IBM.COM
  Key version: 1
  Entry timestamp: 2001/06/08-15:08:12
```

---

You can now obtain a ticket-granting ticket using the key table instead of RACF. Next, issue the **kinit** command to obtain a ticket-granting ticket using the key table, as shown in Example 4-26.

*Example 4-26 Using the kinit command to obtain a ticket-granting ticket using the key table*

---

```
GRAAFF @ SC57:/u/graaff>kinit -k paul
EUVF06014E Unable to obtain initial credentials.
  Status 0x96c73a06 - Client principal is not found in security registry.
```

---

After you run this command, an error indicates that the client principal is not found in the security registry. So, what really happened here? When you look at the trace of the **kinit** command, the issue becomes clear, as shown in Example 4-27.

*Example 4-27 Trace of the kinit command*

---

```
....
kdb_racf_get_principal(): No RACF profile for paul
kdc_as_process_request(): AS_REQ: kdb_get_principal() failed for
paul@KRB390.IBM
kdc_as_process_request(): AS_REQ: KDC error 6 processing request from
paul@KRB390.IBM for krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
```

---

As shown in the trace output, it states no RACF user was found for paul. Local Kerberos principals are always defined in RACF, and foreign principals in their respective REALM (KDC). The messages here indicate that the Kerberos server tried to map the local principal to a RACF user ID and could not find a local principal named paul.

The next step is to define a RACF user ID with a KERB segment and a KERBNAME of paul. We change the RACF user ID GRAAFF to reflect the local Kerberos principal paul, as shown in Example 4-28.

*Example 4-28 Changing the RACF user ID to GRAAFF*

---

```
alu graaff kerb(kerbname(graaff) password(xxx) noexpired
```

---

We try to execute the **kinit -k** command again, as shown in Example 4-29.

*Example 4-29 Issuing the kinit -k command again*

---

```
GRAAFF @ SC57:/u/graaff>kinit -k paul
EUVF06016E Password is not correct for paul@KRB390.IBM.COM.
```

---

The password that we use to add the principal must match the RACF password for the RACF user ID to which it is mapped. So, we must redefine the local principal in the key table using the correct (RACF) password. To redefine the local principal, delete and add the principal as shown in Example 4-30 on page 189.

#### Example 4-30 Redefining the local principal

---

```
GRAAFF @ SC57:/u/graaff>keytab delete paul
GRAAFF @ SC57:/u/graaff>keytab add paul -p racfpw
GRAAFF @ SC57:/u/graaff>klist -k
    Key table: /etc/skrb/krb5.keytab
Principal: paul@KRB390.IBM.COM
    Key version: 1
```

---

We can now issue the **kinit -k paul** command again. Example 4-31 shows that we still receive a password error because we added the principal with the correct password, but using lowercase letters.

#### Example 4-31 Issuing kinit -k paul again

---

```
GRAAFF @ SC57:/u/graaff>kinit -k paul
EUVF06016E Password is not correct for paul@KRB390.IBM.COM.
```

---

Again we need to redefine the principal, as shown in Example 4-30, but we now add the (RACF) password in uppercase. We obtain a ticket-granting ticket successfully, as shown in Example 4-32.

#### Example 4-32 Receiving a ticket-granting ticket successfully

---

```
GRAAFF @ SC57:/u/graaff>kinit -k paul
GRAAFF @ SC57:/u/graaff>klist
    Ticket cache: FILE:/var/skrb/creds/krbcred_b2a52720
    Default principal: paul@KRB390.IBM.COM

Server: krbtgt/KRB390.IBM.COM@KRB390.IBM.COM
Valid 2001/06/15-14:22:42 to 2001/06/16-00:22:42
```

---

## The kadmin command

The **kadmin** command is used to manage entries in the Kerberos database. It prompts you to enter one or more subcommands. The **kadmin** command can be used with any Kerberos administration server supporting Version 2 of the Kerberos administration protocol. The command has the following format:

```
kadmin [-r realm ][-p principal ][-k keytab ][-w password ][-A ][-e ]
```

Where:

- r realm** Specifies the Kerberos administration realm. If this option is not specified, the realm is obtained from the principal name. This option is meaningful only if the administration server supports multiple realms.
- p principal** Specifies the administrator principal. If this option is not specified, the string `/admin` is appended to the principal name obtained from the default credentials cache. If there is no credentials cache, the string `/admin` is appended to the name obtained from the `USER` environment variable, or if the `USER` environment variable is not defined, it is appended to the name obtained from the `getpwuid()` function. The local realm is used if an explicit realm is not part of the principal name.
- k keytab** Specifies the key table that contains the password for the administrator principal. The user is prompted to enter the password if neither the **-k** or the **-w** option is specified. The principal name is host or host name unless the **-p**

option is specified. The host name is the primary host name for the local system.

- w password** Specifies the password for the administrator principal. The user is prompted to enter the password if neither the **-k** nor the **-w** option is specified.
- A** Specifies that the initial ticket used by the **kadmin** command does not contain a list of client addresses. If this option is not specified, the ticket contains the local host address list. When an initial ticket contains an address list, it can be used only from one of the addresses in the address list.
- e** Echoes each command line to **stdout**. This is useful when **stdout** is redirected to a file.

**Note:** Subcommand options start with a minus (-) character and principal attributes start with a plus (+) character or a minus (-) character.

The **kadmin** command imposes no other restrictions on the characters used in names or passwords, although it is recommended that you do not use any of the EBCDIC variant characters. The Kerberos administration server can impose additional restrictions.

### ***Time units***

You can use time units such as dates that are displayed as day-of-week, month, day-of-month, hour:minute:second, time zone, or year using the local time zone, as specified by the **TZ** environment variable. Durations are displayed as days-hours:minutes:seconds.

The **kadmin** command supports a number of date and duration formats and some examples are as follows:

"15 minutes" - "7 days" - "1 month" - "2 hours" - "400000 seconds" - "next year" - "this Monday"

### **Subcommands**

The following subcommand descriptions assume that the administration server is using the standard MIT Kerberos database for the registry. Other database implementations might not support all of the subcommand options and attributes.

### ***Principal-related commands***

**Note:** In the subcommands that we describe in this section, *name* specifies a Kerberos principal.

There is a long list of options that you can use in defining a principal, such as the types of tickets that it can use, what services it can provide, what encryption types are supported for this principal, and what pre-authentication steps might be required.

The following subcommands are supported:

► **help** [*subcommand*]

The **help** subcommand displays the command syntax for the specified subcommand. If no subcommand name is specified, the available subcommands are displayed.

► **get\_privs**

This lists the administrative privileges for the authenticated client.

► **list\_principals** [*expression*]

The **list\_principals** (also known as **listprincs**) subcommand lists all of the principals in the Kerberos database that match the specified search expression. If no search expression is provided, all principals are listed. You must have LIST authority.

► **get\_principal** *name*

The **get\_principal** (also known as **getprinc**) subcommand displays information for a single principal entry. You must have GET authority, or the principal entry must be your own entry.

► **add\_principal** [*options*][*attributes*] *name*

The **add\_principal** (also known as **addprinc**) subcommand adds a new principal entry to the Kerberos database. The options and attributes can be specified before or after the principal name and can be entered in any order. You must have ADD authority.

► **delete\_principal** *name*

The **delete\_principal** (also known as **delprinc**) subcommand deletes a principal entry from the Kerberos database. You must have DELETE authority.

► **modify\_principal** [*options*][*attributes*] *name*

The **modify\_principal** (also known as **modprinc**) subcommand modifies an existing principal entry in the Kerberos database. The options and attributes can be specified before or after the principal name and can be entered in any order. You must have MODIFY authority.

► **change\_password** [-randkey | -pw *password*] [-keepold] [-e *keytypes*] *name*

The **change\_password** (also known as **cpw**) subcommand changes the password for a principal. You must have CHANGE PW authority, or the principal entry must be your own entry.

► **rename\_principal** *oldname newname*

The **rename\_principal** (also known as **renprinc**) subcommand changes the name of a principal entry in the Kerberos database. You must have both ADD and DELETE authority.

### **Policy-related commands**

**Note:** Policy is associated with a password. It specifies characteristics such as the password lifetime, length, number of character classes that must be present, and number of passwords kept in the password history. Passwords in the password history cannot be reused.

► **list\_policies** [*expression*]

The **list\_policies** (also known as **listpols**) subcommand lists all of the policies in the Kerberos database that match the specified search expression. All policies are listed if no search expression is provided. You must have LIST authority.

► **get\_policy** *name*

The **get\_policy** (also known as **getpol**) subcommand displays information for a single policy entry. You must have GET authority or the policy must be associated with your own principal entry.

► **add\_policy** [*options*] *name*

The **add\_policy** (also known as **addpol**) subcommand adds a new policy to the Kerberos database. The options can be specified before or after the policy name and can be specified in any order. You must have ADD authority.

► **modify\_policy** [*options*] *name*

The **modify\_policy** (also known as **modpol**) subcommand modifies an existing policy in the Kerberos database. The options can be specified before or after the policy name and can be specified in any order. You must have MODIFY authority.

► **delete\_policy** *name*

The **delete\_policy** (also known as **delpol**) subcommand deletes a policy entry from the Kerberos database. You must have DELETE authority.

► **add\_key** [[-keytab|-k] *keytab\_name*] [-keepold] [-e *keytypes*] *principal\_name*

The **add\_key** (also known as **ktadd**) subcommand generates a set of random encryption keys for the named principal and then adds the generated keys to the specified key table. The default key table is used if the -keytab option is not specified. A key table name prefix of FILE is changed to FILE because the **add\_key** subcommand must update the key table.

## The kpasswd command

The **kpasswd** command changes the password for a Kerberos principal using the password change service. You must supply the current password for the principal as well as the new password. The password change server applies any applicable password policy rules to the new password before changing the password. The command is issued as follows:

```
kpasswd [principal]
```

The **principal** option specifies the principal whose password is to be changed. The principal is obtained from the default credentials cache if the principal is not specified on the command line.

**Note:** You cannot change the password for a ticket-granting service principal (krbtgt/realm) using the **kpasswd** command.

## The kvno command

The **kvno** command displays the current key version number for a principal and is issued as follows:

```
kvno [principal]
```

The **principal** option specifies the principal whose current key version number is to be displayed. The principal is obtained from the default credentials cache if the principal is not specified on the command line.

Figure 4-18 on page 193 shows success and failure events for auditing.

## 4.16 Auditing

```
KTICKET FAILURE 14:19:08 2001-06-15 /.../KRB390.IBM.COM/paul GRAAFF 24
.....
KTICKET SUCCESS 14:22:42 2001-06-15 /.../KRB390.IBM.COM/paul GRAAFF
```

Figure 4-18 Auditing

### Auditing

SMF Type 80 records are created for login requests (Kerberos initial ticket requests). Both success and failure events can be logged as determined by the `SKDC_LOGIN_AUDIT` environment variable. The event code is 68 and the record includes relocate sections 333 (Kerberos principal name), 334 (request source), and 335 (KDC error code).

The Kerberos principal is stored as a global name (`/.../realm-name/principal-name`) and not as a Kerberos name (`principal-name@realm-name`). This is done to avoid code page problems caused by the at-sign variant character. If the request is received through TCP/IP, the request source is the network address (`nnn.nnn.nnn.nnn:ppppp`). If the request is received through Program Call, the request source is the system user ID of the requester. The KDC error code is a value 0 - 127.

Figure 4-18 shows the smf records (truncated) generated for the kinit commands issued in Example 4-31 on page 189 and Example 4-32 on page 189. The first record shown in Figure 4-18 indicates an error code of 24, which means that the reauthentication (password) failed.

Figure 4-19 on page 194 shows an overview of EIM.

## 4.17 Overview of EIM

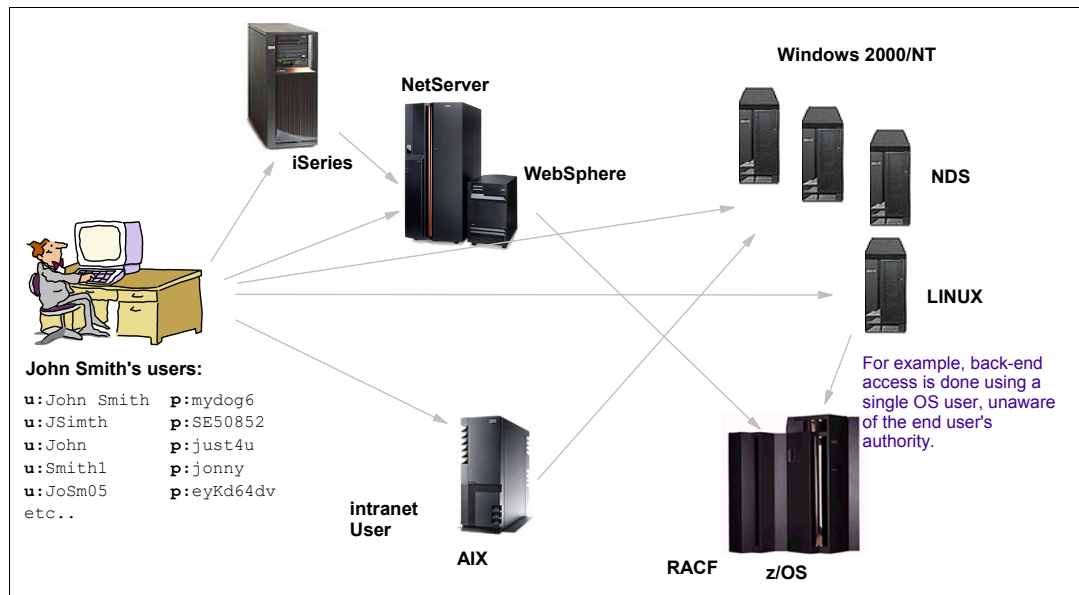


Figure 4-19 Overview of EIM

### Overview of EIM

Today's network environments are made up of a complex group of systems and applications, resulting in the need to manage multiple user registries. Dealing with multiple user registries quickly grows into a large administrative problem that affects users, administrators, and application developers. Consequently, many companies are struggling to securely manage authentication and authorization for systems and applications. Enterprise Identity Mapping (EIM) is an IBM eServer™ infrastructure technology that allows administrators and application developers to address this problem more easily and inexpensively than previously possible.

EIM offers a new approach to enable inexpensive solutions to easily manage multiple user registries and user identities in an enterprise. EIM is an architecture for describing the relationships between individuals or entities (such as file servers and print servers) in the enterprise and the many identities that represent them within an enterprise. In addition, EIM provides a set of APIs that allow applications to ask questions about these relationships.

For example, given a person's user identity in one user registry, you can determine which user identity in another user registry represents that same person. If the user has authenticated with one user identity and you can map that user identity to the appropriate identity in another user registry, the user does not need to provide credentials for authentication again. You know who the user is and only need to know which user identity represents that user in another user registry. Therefore, EIM provides a generalized identity mapping function for the enterprise.

EIM allows one-to-many mappings (that is, a single user with more than one user identity in a single user registry). However, the administrator does not need to have specific individual mappings for all user identities in a user registry. EIM also allows many-to-one mappings (that is, multiple users mapped to a single user identity in a single user registry).

The ability to map between a user's identities in different user registries provides many benefits. Primarily, it means that applications may have the flexibility of using one user



registry for authentication while using an entirely different user registry for authorization. For example, an administrator could map an SAP identity (or better yet, SAP could do the mapping itself) to access SAP resources.

The use of identity mapping requires that administrators do the following tasks:

1. Create EIM identifiers that represent people or entities in their enterprise.
2. Create EIM registry definitions that describe the existing user registries in their enterprise.
3. Define the relationship between the user identities in those registries to the EIM identifiers that they created.
4. Create policy associations.

No code changes are required to existing user registries. The administrator does not need to have mappings for all identities in a user registry. EIM allows one-to-many mappings (that is, a single user with more than one user identity in a single user registry). EIM also allows many-to-one mappings (that is, multiple users sharing a single user identity in a single user registry, which although supported, is not advised). An administrator can represent any user registry of any type in EIM.

EIM is an open architecture that administrators can use to represent identity mapping relationships for any registry. It does not require copying existing data to a new repository and trying to keep both copies synchronized. The only new data that EIM introduces is the relationship information. Administrators manage this data in an LDAP directory, which provides the flexibility of managing the data in one place and having replicas wherever the information is used. On z/OS, an LDAP directory is provided by the product Tivoli Directory Server for z/OS. Ultimately, EIM gives enterprises and application developers the flexibility to easily work in a wider range of environments with less cost than would be possible without this support.

Figure 4-20 on page 196 shows EIM concepts.

## 4.18 EIM concepts

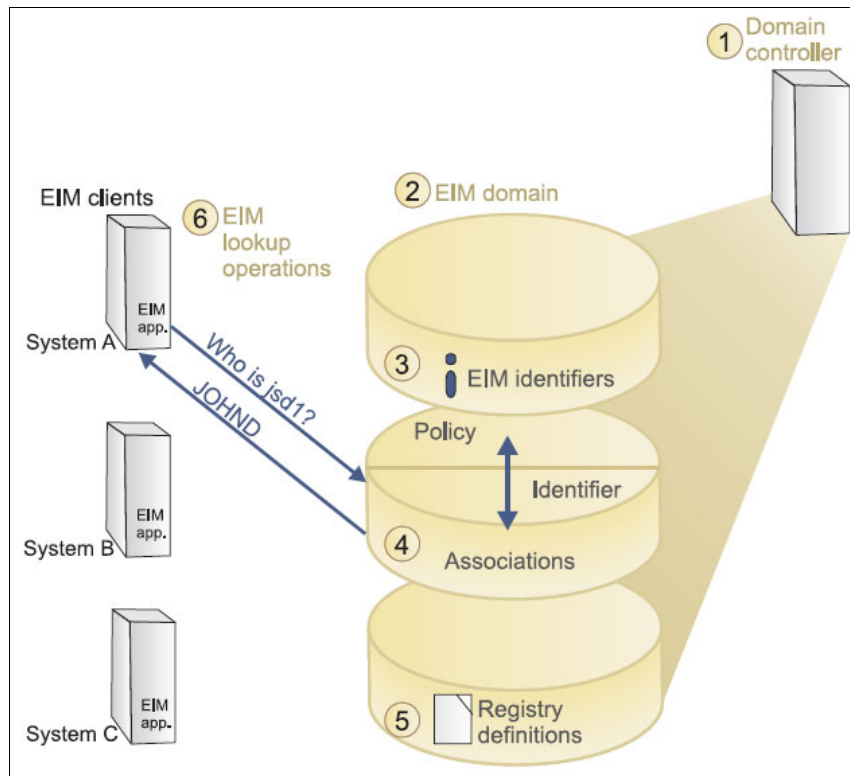


Figure 4-20 EIM concepts

### EIM concepts

A conceptual understanding of how EIM works is necessary to fully understand how you can use EIM in your enterprise. Although the configuration and implementation of EIM APIs can differ among server platforms, EIM concepts are common across IBM eServer servers.

Figure 4-20 provides an EIM implementation example in an enterprise. Three servers act as EIM clients and contain EIM-enabled applications that request EIM data using lookup operations. The domain controller stores information about the EIM domain, which includes an EIM identifier, associations between these EIM identifiers and user identities, and EIM registry definitions.

### EIM domain controller

The EIM domain controller is a Lightweight Directory Access Protocol (LDAP) server that is configured to manage at least one EIM domain. An EIM domain is an LDAP directory that consists of all the EIM identifiers, EIM associations, and user registries that are defined in that domain. Systems (EIM clients) participate in the EIM domain by using the domain data for EIM lookup operations. A minimum of one EIM domain controller must exist in the enterprise.

Currently, you can configure a number of IBM platforms to act as an EIM domain controller. Any system that supports the EIM APIs can participate as a client in the domain. These client systems use EIM APIs to contact an EIM domain controller to perform EIM lookup operations.

The location of the EIM client determines whether the EIM domain controller is a local or remote system. The domain controller is local if the EIM client is running on the same system

as the domain controller. The domain controller is remote if the EIM client is running on a separate system from the domain controller.

## EIM domain

An EIM domain is a directory within an LDAP server that contains EIM data for an enterprise. An EIM domain is the collection of all the EIM identifiers, EIM associations, and user registries that are defined in that domain. Systems (EIM clients) participate in the domain by using the domain data for EIM lookup operations.

An EIM domain is different from a user registry. A user registry defines a set of user identities that are known to and trusted by a particular instance of an operating system or application. A user registry also contains the information needed to authenticate the user of the identity. Additionally, a user registry often contains other attributes such as user preferences, system privileges, or personal information for that identity.

In contrast, an EIM domain refers to user identities that are defined in user registries. An EIM domain contains information about the relationship between identities in various user registries (user name, registry type, and registry instance) and the actual people or entities that these identities represent. Because EIM tracks relationship information only, there is nothing to synchronize between user registries and EIM.

The right side of Figure 4-20 on page 196 shows the data that is stored within an EIM domain. This data includes EIM identifiers, EIM registry definitions, and EIM associations. EIM data defines the relationship between user identities and the people or entities that these identities represent in an enterprise.

EIM data includes:

- ▶ **EIM identifier:** Each EIM identifier that you create represents a person or entity (such as a print server or a file server) within an enterprise.
- ▶ **EIM registry definition:** Each EIM registry definition that you create represents an actual user registry (and the user identity information it contains) that exists on a system within the enterprise. After you define a specific user registry in EIM, that user registry can participate in the EIM domain. You can create two types of registry definitions: one type refers to system user registries, and the other type refers to application user registries.
- ▶ **EIM association:** Each EIM association that you create represents the relationship between an EIM identifier and an associated identity within an enterprise. You must define associations so that EIM clients can use EIM APIs to perform successful EIM lookup operations. These EIM lookup operations search an EIM domain for defined associations between EIM identifiers and user identities in recognized user registries. Associations provide the information that ties an EIM identifier to a specific user identity in a specific user registry.

You can create two different types of associations:

- **Identifier associations:** Identifier associations allow you to define a one-to-one relationship between user identities through an EIM identifier defined for an individual. Each EIM identifier association that you create represents a single, specific relationship between an EIM identifier and an associated user identity within an enterprise.

Identifier associations provide the information that ties an EIM identifier to a specific user identity in a specific user registry and allow you to create one-to-one identity mapping for a user. Identity associations are especially useful when individuals have user identities with special authorities and other privileges that you want to specifically control by creating one-to-one mappings between their user identities.

- **Policy associations:** Policy associations allow you to define a relationship between a group of user identities in one or more user registries and an individual user identity in another user registry. Each EIM policy association that you create results in a many-to-one mapping between the source group of user identities in one user registry and a single target user identity. Typically, you create policy associations to map a group of users who all require the same level of authorization to a single user identity with that level of authorization.

After you create your EIM identifiers, registry definitions, and associations, you can begin using EIM to more easily organize and work with user identities within your enterprise.

## EIM identifier

An EIM identifier represents a person or entity in an enterprise. A typical network consists of various hardware platforms and applications and their associated user registries. Most platforms and many applications use platform-specific or application-specific user registries. These user registries contain all of the user identification information for users who work with those servers or applications.

When you create an EIM identifier and associate it with the various user identities for a person or entity, it becomes easier to build heterogeneous, multitier applications (for example, a single sign-on environment). When you create an EIM identifier and associations, it also becomes easier to build and use tools that simplify the administration involved with managing every user identity that a person or entity has within the enterprise.

## EIM registry definition

An EIM registry definition represents an actual user registry that exists on a system within the enterprise. A user registry operates such as a directory and contains a list of valid user identities for a particular system or application. A basic user registry contains user identities and their passwords. One example of a user registry is the database solely maintained by the z/OS Security Server (commonly referred to as the *RACF database*). User registries can contain other information as well.

For example, an LDAP directory contains bind distinguished names, passwords, and access controls to data that is stored in LDAP. Other examples of common user registries are a Kerberos key distribution center (KDC) and the IBM i user profiles registry.

You can also define user registries that exist within other user registries. Some applications use a subset of user identities within a single instance of a user registry. For example, the z/OS Security Server database can contain specific user registries that are a subset of users within the overall RACF user registry. To model this behavior, EIM allows administrators to create two kinds of EIM registry definitions:

- ▶ System registry definitions
- ▶ Application registry definitions

EIM registry definitions provide information regarding those user registries in an enterprise. The administrator defines these registries to EIM by providing the following information:

- ▶ A unique, arbitrary EIM registry name
- ▶ The type of user registry

Each registry definition represents a specific instance of a user registry. Consequently, you need to choose an EIM registry definition name that helps you to identify the particular instance of the user registry. For example, you could choose the TCP/IP host name for a system user registry, or the host name combined with the name of the application for an application user registry. You can use any combination of alphanumeric characters, mixed case, and spaces to create unique EIM registry definition names.

There are a number of predefined user registry types that EIM provides to cover most operating system user registries, including:

- ▶ IBM AIX®
- ▶ IBM Domino® - long name
- ▶ Domino - short name
- ▶ Kerberos
- ▶ Kerberos - case sensitive
- ▶ LDAP
- ▶ Linux
- ▶ Policy director
- ▶ Novell Directory Server
- ▶ IBM i (previously known as OS/400® or i5/OS)
- ▶ Tivoli Access Manager
- ▶ z/OS Security Server
- ▶ Windows - local
- ▶ Windows domain (Kerberos)
- ▶ X.509

**Note:** Although the predefined registry definition types cover most operating system user registries, you might need to create a registry definition for which EIM does not include a predefined registry type. You have two options in this situation. You can either use an existing registry definition that matches the characteristics of your user registry, or you can define a private user registry type.

For example, in Figure 4-21 on page 200, the administrator followed the process required and defined the type of registry as WebSphere Third-Party Authentication (LTPA) for the System\_A\_WAS application registry definition.

In Figure 4-21 on page 200, the administrator creates EIM registry definitions for user registries representing System A, System B, and System C and a Windows Active Directory that contains users' Kerberos principals with which users log in to their desk top workstations. In addition, the administrator created an application registry definition for WebSphere Lightweight Third Party Authentication (LTPA), which runs on System A.

The registry definition name that the administrator uses helps to identify the specific occurrence of the type of user registry. For example, an IP address or host name is often sufficient for many types of user registries. In this example, the administrator identifies the specific user registry instance by using System\_A\_WAS as the registry definition name to identify this specific instance of the WebSphere LTPA application. In addition to the name, the administrator also provides the type of registry as System\_A.

Figure 4-21 on page 200 shows EIM registry definitions.

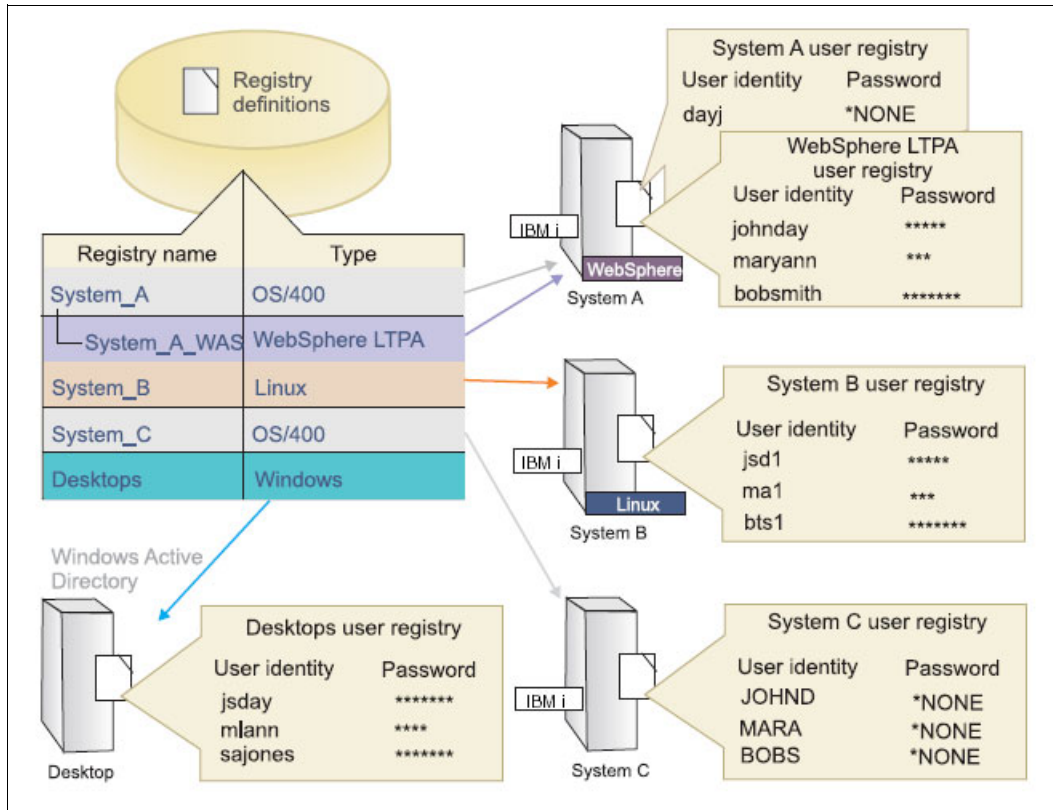


Figure 4-21 EIM registry definitions

You can also define user registries that exist within other user registries. For example, the z/OS Security Server (RACF) registry can contain specific user registries that are a subset of users within the overall RACF user registry. A RACF user profile can contain an EIM segment, which points to a profile in the LDAPBIND class, this latter profile contains the name of the EIM Domain and bind information needed to establish a connection with the EIM Domain.

## EIM associations

An EIM association is an entry that you create in an EIM domain to define a relationship between user identities in different user registries. The type of association that you create determines whether the defined relationship is direct or indirect. You can create one of two types of associations in EIM: identifier associations and policy associations. You can use policy associations instead of, or in combination with, identifier associations. How you use associations depends on your overall EIM implementation plan.

## EIM lookup operation

An application or an operating system uses an EIM API to perform a lookup operation so that the application or operating system can map from one user identity in one registry to another user identity in another registry. An EIM lookup operation is a process through which an application or operating system finds an unknown associated user identity in a specific target registry by supplying some known and trusted information. Applications that use EIM APIs can perform these EIM lookup operations on information only if that information is stored in the EIM domain. An application can perform one of two types of EIM lookup operations based on the type of information the application supplies as the source of the EIM lookup operation: a user identity or an EIM identifier.

When applications or operating systems use the `eimGetTargetFromSource` API to obtain a target user identity for a given target registry, they must supply a user identity as the source of the lookup operation. To be used as the source in an EIM lookup operation, a user identity must have either an identifier source association defined for it or be covered by a policy association.

When an application or operating system uses this API, the application or operating system must supply these pieces of information:

- ▶ A user identity as the source or starting point of the operation.
- ▶ The EIM registry definition name for the source user identity.
- ▶ The EIM registry definition name that is the target of the EIM lookup operation. This registry definition describes the user registry that contains the user identity that the application is seeking.

When applications or operating systems use the `eimGetTargetFromIdentifier` API to obtain a user identity for a given target registry, they must supply an EIM identifier as the source of the EIM lookup operation. When an application uses this API, the application must supply the following pieces of information:

- ▶ A user identity as the source, or starting point of the operation.
- ▶ The EIM registry definition name that is the target of the EIM lookup operation. This registry definition describes the user registry that contains the user identity that the application is seeking.

Within z/OS, the calling application that is using the `eimGetTargetFromIdentifier` API can be running in system key or supervisor state, or:

- ▶ The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class.
- ▶ The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class.
- ▶ And the FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.

For a user identity to be returned as the target of either type of EIM lookup operation, the user identity must have a target association defined for it. This target association can be in the form of an identifier association or a policy association.

The supplied information is passed to EIM and the lookup operation searches for and returns any target user identities, by searching EIM data in the following order:

1. Identifier target association for an EIM identifier. The EIM identifier is identified in one of two ways: It is supplied by the `eimGetTargetFromIdentifier` API. Alternatively, the EIM identifier is determined from information supplied by the `eimGetTargetFromSource` API.
2. Certificate filter policy association.
3. Default registry policy association.
4. Default domain policy association.

Figure 4-22 on page 202 shows the EIM lookup operation.

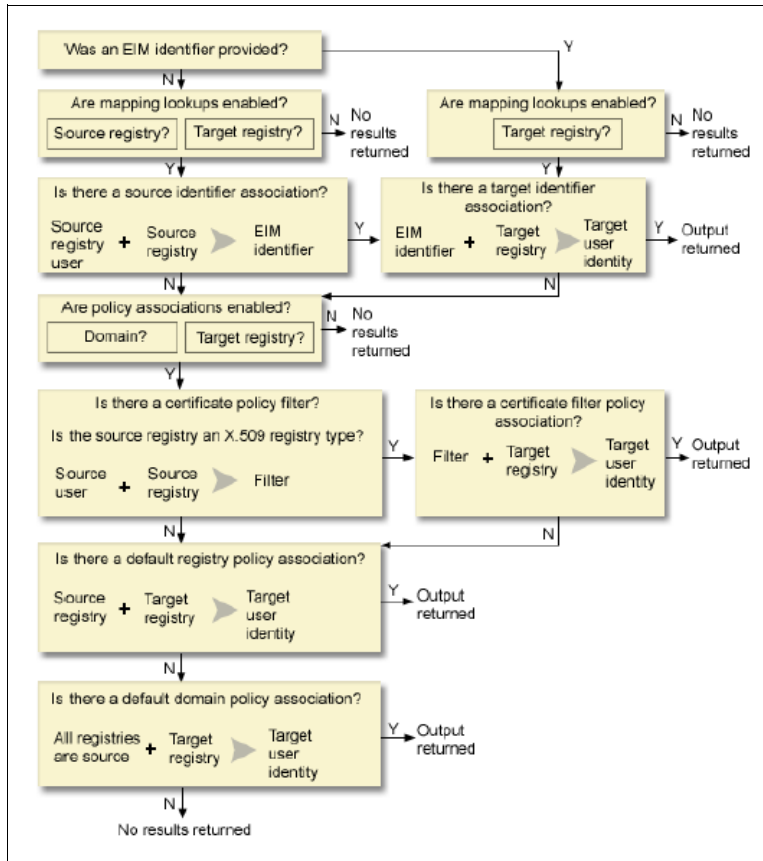


Figure 4-22 EIM lookup operation

The lookup operation, illustrated in Figure 4-22, searches flows in this manner:

1. The lookup operation checks whether mapping lookups are enabled. The lookup operation determines whether mapping lookups are enabled for the specified source registry, the specified target registry, or both specified registries. If mapping lookups are not enabled for one or both of the registries, the lookup operation ends without returning a target user identity.
2. The lookup operation checks whether there are identifier associations that match the lookup criteria. If an EIM identifier was provided, the lookup operation uses the specified EIM identifier name. Otherwise, the lookup operation checks whether there is a specific identifier source association that matches the supplied source user identity and source registry. If there is one, the lookup operation uses it to determine the appropriate EIM identifier name. The lookup operation then uses the EIM identifier name to search for an identifier target association for the EIM identifier that matches the specified target EIM registry definition name. If there is an identifier target association that matches, the lookup operation returns the target user identity defined in the target association.
3. The lookup operation checks whether the use of policy associations are enabled. The lookup operation checks whether the domain is enabled to allow mapping lookups using policy associations. The lookup operation also checks whether the target registry is enabled to use policy associations. If the domain is not enabled for policy associations or the registry is not enabled for policy associations, the lookup operation ends without returning a target user identity.
4. The lookup operation checks for certificate filter policy associations. The lookup operation checks whether the source registry is an X.509 registry type. If it is an X.509 registry type, the lookup operation checks whether there is a certificate filter policy association that



matches the source and target registry definition names. The lookup operation checks whether there are certificates in the source X.509 registry that satisfy the criteria specified in the certificate filter policy association. If there is a matching policy association and there are certificates that satisfy the certificate filter criteria, the lookup operation returns the appropriate target user identity for that policy association.

5. The lookup operation checks for default registry policy associations. The lookup operation checks whether there is a default registry policy association that matches the source and target registry definition names. If there is a matching policy association, the lookup operation returns the appropriate target user identity for that policy association.
6. The lookup operation checks for default domain policy associations. The lookup operation checks whether there is a default domain policy association defined for the target registry definition. If there is a matching policy association, the lookup operation returns the associated target user identity for that policy association.
7. The lookup operation is unable to return any results.

When an application supplies a user identity as the source, the application also must supply the EIM registry definition name for the source user identity and the EIM registry definition name that is the target of the EIM lookup operation. To be used as the source in a EIM lookup operation, a user identity must have a source association defined for it.

When an application supplies an EIM identifier as the source of the EIM lookup operation, the application must also supply the EIM registry definition name that is the target of the EIM lookup operation. For a user identity to be returned as the target of either type of EIM lookup operation, the user identity must have a target association defined for it.

The supplied information is passed to the EIM domain controller, where all EIM information is stored and the EIM lookup operation searches for the source association that matches the supplied information. Based on the EIM identifier (supplied to the API or determined from the source association information), the EIM lookup operation then searches for a target association for that identifier that matches the target EIM registry definition name.

In Figure 4-23, the user identity johnday authenticates to the WebSphere Application Server using LTPA on System A.

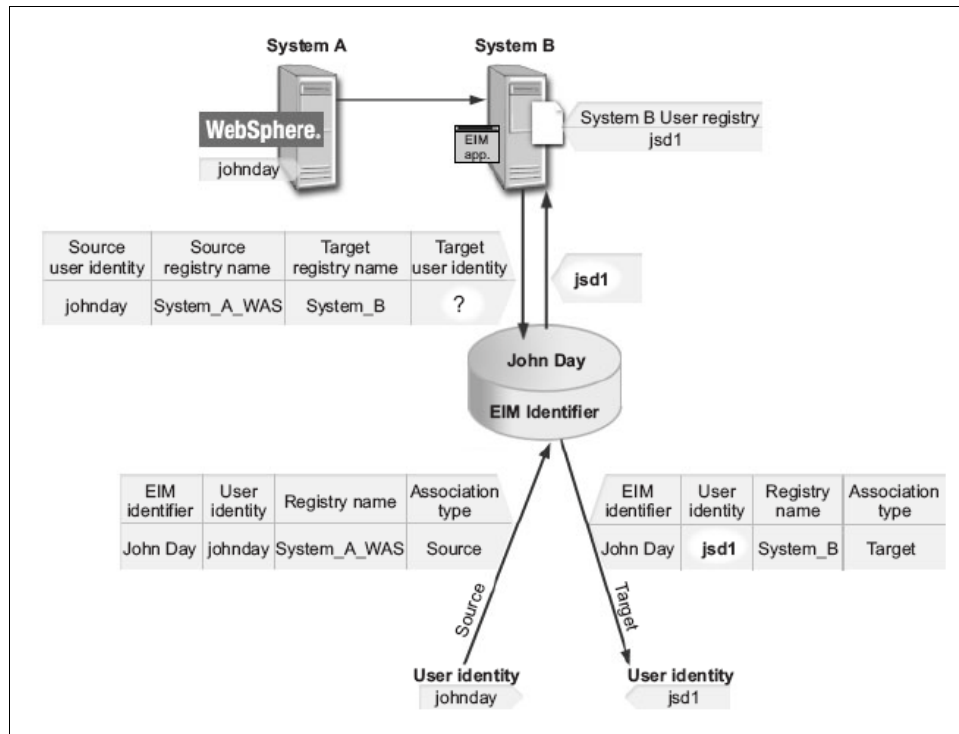


Figure 4-23 EIM lookup

The WebSphere Application Server on System A calls a native program on System B to access data on System B. The native program uses an EIM API to perform an EIM lookup operation based on the user identity on System A as the source of the operation. The application supplies the following information to perform the operation:

- ▶ johnday as the source user identity
- ▶ System\_A\_WAS as the source EIM registry definition name
- ▶ System\_B as the target EIM registry definition name

This source information is passed to the EIM domain controller and the EIM lookup operation finds a source association that matches the information. Using the EIM identifier name, the EIM lookup operation searches for a target association for the johnday identifier that matches the target EIM registry definition name for System\_B. When the matching target association is found, the EIM lookup operation returns the jsd1 user identity to the application.

### Mapping policy support and enablement

EIM mapping policy support allows you to use policy associations as well as specific identifier associations in an EIM domain. You can use policy associations instead of, or in combination with, identifier associations.

EIM mapping policy support provides a means of enabling and disabling the use of policy associations for the entire domain, as well as for each specific target user registry. EIM also allows you to set whether a specific registry can participate in mapping lookup operations in general. Consequently, you can use mapping policy support to more precisely control how mapping lookup operations return results.

The default setting for an EIM domain is that mapping lookups that use policy associations are disabled for the domain. When the use of policy associations is disabled for the domain, all mapping lookup operations for the domain return results only by using specific, identifier associations between user identities and EIM identifiers.

The default setting for each individual registry is that mapping lookup participation is enabled and the use of policy associations is disabled. When you enable the use of policy associations for an individual target registry, you must also ensure that this setting is enabled for the domain.

You can configure mapping lookup participation and the use of policy associations for each registry in one of the following ways:

- ▶ Mapping lookup operations cannot be used for the specified registry at all. In other words, an application that performs a mapping lookup operation involving that registry will fail to return results.
- ▶ Mapping lookup operations can use specific identifier associations between user identities and EIM identifiers only. Mapping lookups are enabled for the registry, but the use of policy associations is disabled for the registry.
- ▶ Mapping lookup operations can use specific identifier associations when they exist and policy associations when specific identifier associations do not exist (all settings are enabled).

## **EIM access control**

An EIM user is a user who possesses EIM access control based on their membership in a predefined LDAP user group for a specific domain. Specifying EIM access control for a user adds that user to a specific LDAP user group for a particular domain. Each LDAP group has authority to perform specific EIM administrative tasks for that domain. Which and what type of administrative tasks, including lookup operations, that an EIM user can perform is determined by the access control group to which the EIM user belongs.

EIM access controls allow a user to perform specific administrative tasks or EIM lookup operations. Only users with EIM administrator access are allowed to grant or revoke authorities for other users. EIM access controls are granted only to user identities that are known to the EIM domain controller.

The following sections provide brief descriptions of the functions that each EIM access control group can perform.

### ***LDAP administrator***

This access control allows the user to configure a new EIM domain. A user with this access control can perform the following functions:

- ▶ Create and delete a domain
- ▶ Create and remove EIM identifiers
- ▶ Create and remove EIM registry definitions
- ▶ Create and remove source, target, and administrative associations
- ▶ Perform EIM lookup operations
- ▶ Retrieve associations, EIM identifiers, and EIM registry definitions
- ▶ Add, remove, and list EIM authority information

### ***EIM administrator***

This access control allows the user to manage all of the EIM data within this EIM domain. A user with this access control can perform the following functions:

- ▶ Delete a domain

- ▶ Create and remove EIM identifiers
- ▶ Create and remove EIM registry definitions
- ▶ Create and remove source, target, and administrative associations
- ▶ Perform EIM lookup operations
- ▶ Retrieve associations, EIM identifiers, and EIM registry definitions
- ▶ Add, remove, and list EIM authority information

### ***EIM identifiers administrator***

This access control allows the user to add and change EIM identifiers and manage source and administrative associations. A user with this access control can perform the following functions:

- ▶ Create an EIM identifier
- ▶ Add and remove source associations
- ▶ Add and remove administrative associations
- ▶ Perform EIM lookup operations
- ▶ Retrieve associations, EIM identifiers, and EIM registry definitions

### ***EIM mapping lookup***

This access control allows the user to conduct EIM lookup operations. A user with this access control can perform the following functions:

- ▶ Perform EIM lookup operations
- ▶ Retrieve associations, EIM identifiers, and EIM registry definitions

### ***EIM registries administrator***

This access control allows the user to manage all EIM registry definitions. A user with this access control can perform the following functions:

- ▶ Add and remove target associations
- ▶ Perform EIM lookup operations
- ▶ Retrieve associations, EIM identifiers, and EIM registry definitions

### ***EIM registry X administrator***

This access control allows the user to manage a specific EIM registry definition. Membership in this access control group also allows the user to add and remove target associations only for a specified user registry definition. To take full advantage of mapping lookup operations and policy associations, a user with this access control should also have EIM mapping operations access control. This access control allows a user to:

- ▶ Create, remove, and list target associations for the specified EIM registry definitions only
- ▶ Add and remove default domain policy associations
- ▶ Add and remove policy associations for the specified registry definitions only
- ▶ Add certificate filters for the specified registry definitions only
- ▶ Enable and disable mapping lookups for the specified registry definitions only
- ▶ Add and remove policy associations only for the specified registries
- ▶ Retrieve EIM identifiers
- ▶ Retrieve identifier associations and certificate filters for the specified registry definitions only
- ▶ Add and remove target associations for the specific EIM registry definition
- ▶ Perform EIM lookup operations
- ▶ Retrieve EIM registry definition information for the specified registry definitions only

Figure 4-24 on page 207 shows setting up an EIM configuration that involves z/OS.

## 4.19 Setting up EIM in z/OS

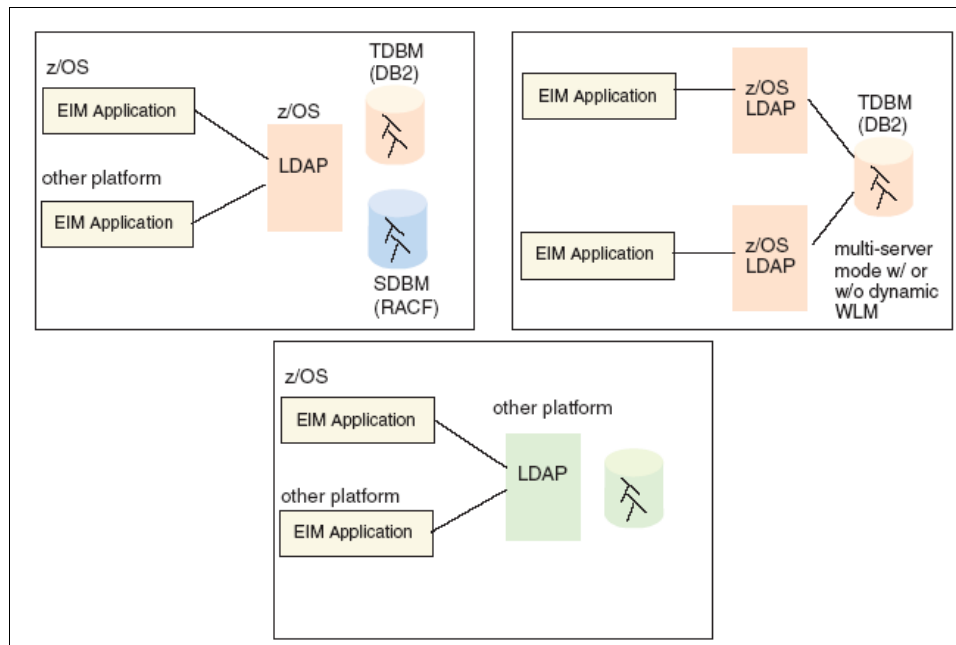


Figure 4-24 Setting up EIM configuration involving z/OS

### Steps for installing and configuring the EIM domain controller on z/OS

**Note:** For the Tivoli Directory Server for z/OS, the following requirements must be met:

- ▶ Tivoli Directory Server must be configured to use the TDBM back end. A TDBM is a general purpose back end that can store any type of directory information. It uses a DB2 database to do this. If a TDBM is not used, an LDBM must be used, which is a z/OS UNIX System Services file system. This is a file-based back end to store directory information.
- ▶ An optional back end called the SDBM, which is the RACF database.

1. Install and configure Tivoli Directory Server:
  - a. Tivoli Directory Server must be configured to accept the different types of bind requests. The best reference for these actions is in the *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-6788.
  - b. Start Tivoli Directory Server.
  - c. Load the schema definitions.

**Attention:**

- ▶ An EIM domain must be updated using the EIM APIs or administrative applications that use the EIM APIs. We do not recommend using the LDAP utilities and LDAP client APIs to update information in an EIM domain.
- ▶ Do not alter the EIM schema definitions unless directed to do so by your IBM service representative during problem diagnosing.

**Restriction:** z/OS Tivoli Directory Server by default has a 511 character limit on the length of a distinguished name for an entry. If this default length is exceeded, message ITY0023 (indicating an unexpected LDAP error) is issued, indicating that DB2 needs to be reconfigured to support longer distinguished names. This error might show up when working with long identifiers, registries, domain names, or suffixes.

2. Consider the options that you have for setting up an EIM domain that includes z/OS:
  - a. Use Tivoli Directory Server on z/OS as the domain controller. (z/OS and non-z/OS applications could access the data.) The Tivoli Directory Server on z/OS must be configured with the TDBM back end. If you plan to use RACF user IDs and passwords for the bind credentials, configure the server with the SDBM and the TDBM back ends.
  - b. Set up the Tivoli Directory Server on z/OS in multi-server mode. This configuration has multiple LDAP servers sharing the TDBM back end store, which is useful if you want to balance the work load between your LDAP servers.
  - c. The z/OS EIM application can access a domain controller that is on another platform.

## 4.20 Installing and configuring EIM on z/OS

### Installing and configuring EIM on z/OS

By way of background, EIM is installed into an HFS or z/OS file system (zFS) directory using z/OS SMP/E with the default directory being /usr/lpp/eim. Figure 4-25 shows the various directories the EIM features are installed into with the default directories listed in the rightmost column.

Directory and description	Default value or customized value
Main install directory	/usr/lpp/eim
EIM library directory; contains the runtime library(eim.dll) and the definition side deck file(eim.x) for linking EIM applications <b>Note:</b> Note: These files are also symbolically linked in the /usr/lib directory.	/usr/lpp/eim/lib
Message catalog directories <b>Note:</b> Files in C directory are symbolically linked to the En_US.IBM-1047 directory message catalog files. There are additional symlinks of the En_US.IBM-1047 message catalog files in the /usr/lib/nls/msg/C and /usr/lib/nls/msg/En_US.IBM-1047 directories. Additionally, there are symbolic links to the message catalog files in the Ja_JP directory in the /usr/lib/nls/msg/Ja_JP directory	/usr/lpp/eim/lib/nls/msg/En_US.IBM-1047 , /usr/lpp/eim/lib/nls/msg/C, and /usr/lpp/eim/lib/nls/msg/Ja_JP
C/C++ header files for the EIM API prototypes, defined data types, and message catalog constants <b>Note:</b> The header files are also symbolically linked in the /usr/include directory.	/usr/lpp/eim/include
EIM programs directory (which is where the eimadmin utility program is located)	/usr/lpp/eim/bin
EIM man page directory <b>Note:</b> There is a symbolic link to the man page in the /usr/man/C/cat1 and /usr/man/En_US.IBM-1047/cat1 directories..	/usr/lpp/eim/man/En_US.IBM-1047/cat1 and /usr/lpp/eim/man/C/cat1

Figure 4-25 Installing and configuring EIM on z/OS

Figure 4-25 lists important directories for EIM installation. Your system programmer should review the rightmost column of this table, crossing out any defaults that have changed and recording the correct directory names.

**Tip:** An EIM administrator who uses the `eimadmin` utility might want that the directory for the `eimadmin` utility be placed in the PATH environment variable. This enables the ability to run the utility without having to specify the path when issuing the command (or changing to the /usr/lpp/eim/bin directory before issuing the command). The PATH environment variable can be modified to include the EIM programs directory by issuing the following command from a shell prompt:

```
export PATH=$PATH:/usr/lpp/eim/bin
```

This adds the EIM program's directory to the end of the list of directories to search for programs. Add the export command to a user's .profile file so that each time the user enters a shell, the PATH is updated.

## Steps for using the eimadmin utility to manage an EIM domain

Perform the steps listed in this section to create and manage an EIM domain using the `eimadmin` utility.

Before you begin:

- ▶ The `eimadmin` utility examples can be entered from the z/OS UNIX System Services shell by an EIM administrator.
- ▶ For improved readability, each command option is shown on a separate line.
- ▶ In most cases, you specify multiple options on a single line, separating them with one or more spaces.
- ▶ If necessary, you can use the backslash (`\`) continuation character to break the command into multiple lines.
- ▶ The access authority required for successful completion depends on the particular `eimadmin` operation you specify, and is determined by the bind credential you specify for LDAP authentication. The distinguished name that LDAP associates with the credential should be a member of one or more EIM access groups, which define access authority to EIM data.

To create the domain, follow these steps:

1. Create an EIM domain by entering a command such as the following from the z/OS shell:

```
eimadmin -aD -d domainDN -n description -h ldapHost -b bindDN -w bindPassword
```

The `bindDN` must be the distinguished name for the LDAP administrator. (The description is optional.)

The following command creates the EIM domain `My Domain`:

```
eimadmin
-aD
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-n 'An EIM Domain'
-h ldap://some.ldap.host
-b 'cn=ldap administrator'
-w secret
```

**Note:** This assumes that the `o=IBM,c=US` objects are defined in the LDAP Directory.

2. Give an EIM administrator authority to the domain by entering a command such as the following command from the z/OS shell:

```
eimadmin
-aC
-d domainDN
-c ADMIN
-q accessUser
-f accessUserType
-h ldapHost
-b bindDN
-w bindPassword
```

The parameter following `-c` is the `accessType` parameter. In this situation, the value must be `ADMIN`. The `bindDN` must be the distinguished name for the LDAP administrator.



**Tip:** If you plan on dividing the administration responsibilities, repeat this command for the other administrative users.

The following command can be issued by the LDAP administrator to give the EIM administrator, `cn=eim administrator,ou=dept20,o=IBM,c=US`, authority to administer the EIM domain:

```
eimadmin
-aC
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-c ADMIN
-q 'cn=eim administrator,ou=dept20,o=IBM,c=US'
-f DN
-h ldap://some.ldap.host
-b 'cn=ldap administrator'
-w secret
```

**Note:** This assumes that the `cn=eim administrator,ou=dept20,o=IBM,c=US` is defined in the LDAP Directory.

3. Add registries to the EIM domain by entering a command such as the following command from the z/OS shell:

```
eimadmin
-aR
-d domainDN
-r registryName
-y registryType
-n description
-h ldapHost
-b bindDN
-w bindPassword
```

**Note:** The `-y` parameter specifies registry type.

The following command adds a RACF registry to the EIM domain named `My Domain`:

```
eimadmin
-aR
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-r 'RACF Pok1'
-y RACF
-n 'the RACF Registry on Pok System 1'
-h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=IBM,c=US'
-w secret
```

The following command adds an IBM i registry to the EIM domain named `My Domain`:

```
eimadmin
-aR
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-r 'OS400 RCH1'
-y OS400
-n 'the OS400 Registry on Rochester System 1'
-h ldap://some.ldap.host
```

```
-b 'cn=eim administrator,ou=dept20,o=IBM,c=US'  
-w secret
```

4. Add enterprise identifiers to the domain by entering a command such as the following from the z/OS shell:

```
eimadmin  
-aI  
-d domainDN  
-i identifier  
-n description  
-h ldapHost  
-b bindDN  
-w bindPassword
```

You can add identifiers at any time after creating the domain.

The preceding command adds a single identifier to the domain. Alternately, you can add multiple identifiers by specifying a file name as standard input to the `eimadmin` utility. Specifying a file name indicates using the file of identifiers as input for batch processing of multiple identifiers.

Repeat this step as needed.

The `bindDN` must have EIM administrator authority or EIM Identifier administrator authority. The following command can be issued by the EIM administrator to add an EIM identifier to the domain `My Domain`:

```
eimadmin  
-aI  
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'  
-i 'John Adam Day'  
-h ldap://some.ldap.host  
-b 'cn=eim administrator,ou=dept20,o=IBM,c=US'  
-w secret
```

5. Create associations between registry user IDs and identifiers by entering commands from the z/OS shell (One or more of the association types, `-t source`, `-t target`, `-t admin` are required on the command.):

```
eimadmin  
-aA  
-d domainDN  
-r registryName  
-u userid  
-i identifier  
-t admin  
-t source  
-t target  
-h ldapHost  
-b bindDN  
-w bindPassword
```

The following command creates associations between the user ID `JD` in the RACF `Pok1` registry:

```
eimadmin  
-aA  
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'  
-r 'RACF Pok1'  
-u JD  
-i 'John Day'
```

```
-t source
-t target
-h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=IBM,c=US'
-w secret
```

After you enter these commands, you can use the domain for lookup operations. For the preceding examples, the only user mappings available are mappings from JD to JOHNDAY and from JOHNDAY to JD.

**Note:** You can create associations only after registries and identifiers are in place.

The command creates only two associations. Conversely, you can create multiple associations by specifying a file name as standard input to the `eimadmin` command. Specifying a file name indicates using a file of associations as input for batch processing of multiple associations.

Repeat this step as needed.

6. Give users lookup access to the EIM domain. Use the following command:

```
eimadmin
-aC
-d domainDN
-c MAPPING
-q accessUser
-f DN
-h ldapHost
-b bindDN
-w bindPassword
```

The `eimadmin` utility allows you to grant access one user at a time or a list of users can be provided in a file using the following command:

```
eimadmin
-aC
-d domainDN
-c MAPPING
-h ldapHost
-b bindDN
-w bindPassword <input-fileName
```

The file must contain a label line following by at least one user name. For example, a bind distinguished name, and the type of the user name as follows:

```
CU ;CS ; cn=John Day,c=US DN
```

The EIM administrator can issue the following command to give the user John Day mapping (lookup) authority to the domain My Domain:

```
eimadmin
-aC
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-c MAPPING
-q 'cn=John Day,c=US' -h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=IBM,c=US'
-w secret
```

Figure 4-26 on page 214 shows domain authentication methods.

## 4.21 Domain authentication methods

- z/OS EIM supports the following authentication methods recognized by LDAP:
  - Simple (with or without CRAM-MD5 password protection)
  - Digital certificate
  - Kerberos

Figure 4-26 Domain authentication methods

### Domain authentication methods

Authentication occurs when an EIM application connects (binds) to the EIM domain controller. z/OS EIM supports the following three authentication methods recognized by LDAP:

- ▶ Simple (with or without CRAM-MD5 password protection)
- ▶ Digital certificate
- ▶ Kerberos

Your LDAP server configuration and security requirements determine which method you choose. The examples in this section illustrate how you can use these methods with the `eimadmin` utility.

This information explains how the bind credentials specified correspond to the distinguished name that LDAP uses for access checking. Your access to EIM data is determined by the authority groups of which the distinguished name is a member. The exception is the distinguished name for the LDAP administrator that has unrestricted access.

### Using simple binds

A distinguished name and password are sufficient credentials for a SIMPLE `eimadmin` connect type, as follows:

```
eimadmin
-1D
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-S SIMPLE
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

**Note:** Unless an SSL session has been established, the password is sent over the network in plain text, making this method the least secure. The distinguished name that you specify is the one LDAP uses for access checking.

### Using CRAM-MD5 password protection

You can use CRAM-MD5 for simple authentication without sending the bind password over the network in plain text, provided both client and server support the method. In the utility command, specify the connect type CRAM-MD5 to indicate simple authentication with password protection, as follows:

```
eimadmin
-1D
```

```
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-S CRAM-MD5  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

## Using digital certificates

To bind using a digital certificate, specify the EXTERNAL connect type on the **eimadmin** command. Ensure that the host name identifies a secure host:port value prefixed with `ldaps://`, as follows:

```
eimadmin  
-lD  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldaps://secure.ldap.host  
-S EXTERNAL  
-K client.kdb  
-P clientpw  
-N eimadmincert
```

**Note:** LDAP uses the client certificate's subject distinguished name for access checking.

Use the following rules:

- ▶ You must also specify the name of either a key database file or RACF key ring that contains your client certificate.
- ▶ You must specify the label for that certificate if it is not the defined default. If you specify a key database file but not its password, the utility prompts you for it.

## Using Kerberos

To bind using a Kerberos identity, specify connect type GSSAPI on the **eimadmin** command. No other credential information is required, but the default Kerberos credential must be established through a service such as **kinit** before entering the command, as follows:

```
kinit eimadministrator@realm.com  
eimadmin  
-lD  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-S GSSAPI
```

For access checking, LDAP considers a distinguished name formed by prefixing the Kerberos principal name with `ibm-kgn=` or distinguished names located through special mapping or searches.

## Using Secure Sockets Layer

You can establish a Secure Sockets Layer (SSL) connection along with any of the supported authentication types if your domain controller is configured as a secure host enabled for server authentication.

A secure host is required for EXTERNAL connect.

The strength of SSL is that data transferred over the connection is encrypted, including the password for a SIMPLE bind. The **eimadmin** utility recognizes the need for an SSL connection

when you specify an LDAP host name prefixed with `ldaps://`. It then requires that you specify a RACF key ring, or a key database file and its password.

Figure 4-27 shows EIM additional administration tasks.

## 4.22 EIM additional administration tasks

- EIM additional administration tasks:
  - Managing registries
    - Adding a system and application registry
    - Removing a registry
  - Assigning an alias
    - Assigning an alias
    - Removing an alias
    - Assigning an alias to a different registry
  - Adding a new user
    - Adding an identifier
    - Adding associations
  - Removing a user
    - Removing associations
    - Removing an identifier
  - Changing access authority
    - Adding access
    - Removing access

Figure 4-27 EIM additional administration tasks

### Managing registries

A domain typically contains multiple registries. User identities for a particular system are associated with a system registry, while a subset of identities might be associated with an application registry.

#### ***Adding a system and application registry***

Create a system registry by entering the following command:

```
eimadmin
-aR
-r 'RACF Pok1'
-y racf
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

Enter the following command to define an application registry that depends on a previously defined system registry:

```
eimadmin
-aR
-r 'App1'
-y racf
-g 'RACF Pok1'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

**Note:** After you define an application registry, you can refer to it by name in EIM APIs and `eimadmin` commands without having to identify it as an application-type registry.

### ***Listing a registry***

You can list any registry using a command similar to the following:

```
eimadmin
-lR
-r 'App1'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

### ***Removing a registry***

To remove a registry, issue the following command:

```
eimadmin
-pR
-r 'App1'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

All associations linked to the registry are deleted automatically.

**Attention:** EIM refuses to remove a system registry if any application registries depend on it.

You can find the dependents that you must remove by searching for all occurrences of the system registry name in the output from the following command, which lists all registries:

```
eimadmin
-lR
-r '*'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

With caution, you can use the `-s rmdeps` option of `eimadmin` to remove dependent application registries automatically when removing the system registry, as follows:

```
eimadmin
-s rmdeps
-pR
-r 'RACF Pok1'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

### Working with registry aliases

You can define alias names to facilitate registry administration. By establishing aliases that applications use to look up actual registry names, you can make nondisruptive registry changes by managing alias assignments.

**Rule:** When defining or referencing a registry alias, you must specify an associated registry type.

### Assigning an alias

Enter the following command to assign an alias name to an existing registry:

```
eimadmin
-mR
-r 'RACF Test Pok1'
-x 'z/OS'
-z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

This example defines the alias `z/OS` (of type `RACF`) for registry `RACF Test Pok1`.

### Listing an alias

You can list the registry and its aliases using the following command:

```
eimadmin
-lR
-r 'RACF Test Pok1'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

### Removing an alias

You can delete an alias for a registry using the following command:

```
eimadmin
-eR
-r 'RACF Test Pok1'
-x 'z/OS'
-z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```



This example removes the alias `z/OS` (of type `RACF`) for registry `RACF Test Pok1`.

### ***Assigning an alias name to a different registry***

To assign an alias name to a different registry, add the alias name and type to the registry attributes as shown in the example for adding an alias name to a registry above. Multiple registries can have the same registry alias values. However, if you want the alias to map to a single registry, you must remove that alias from registries in which it was previously defined.

Enter the following two commands to reassign alias `z/OS` from registry `RACF Test Pok1` to registry `RACF Pok1`:

```
eimadmin
-mR
-r 'RACF Pok1'
-x 'z/OS'
-z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

```
eimadmin
-eR
-r 'RACF Test Pok1'
-x 'z/OS'
-z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

### **Adding a new user**

You can create a new EIM identifier to represent a new person entering your enterprise. As the person is given access to each system or application through its user registry, you can define an EIM association between the EIM identifier and the corresponding registry defined in EIM.

### ***Adding an identifier***

When you create a new EIM identifier, it is assigned a name that is unique within the domain.

The `eimadmin` utility requires that you specify a unique name (unlike the `eimAddIdentifier` API option that generates a unique name for you).

You can assign an alternate name, or alias, to multiple identifiers. This non-unique name can be used to further describe the represented individual or to serve as an alternative identifier for lookup operations.

Enter the following command to add a new identifier `John S. Day` with two aliases:

```
eimadmin
-aI
-i 'John S. Day'
-j '654321'
-j 'Contractor'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
```

```
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

You can list the new identifier using the unique name.

The utility returns one entry only, as follows:

```
eimadmin  
-lI  
-i 'John S. Day'  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

You can also list the new identifier using an alias name.

The utility returns all entries having Contractor defined as an alternative name, as follows:

```
eimadmin  
-lI  
-j 'Contractor'  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

### ***Adding associations***

You can register the system and application user IDs assigned to the individual by defining EIM associations between the identifier and the corresponding registries.

Enter the following command to create source and target associations for user ID JD in registry RACF Pok1:

```
eimadmin  
-aA  
-i 'John S. Day'  
-r 'RACF Pok1'  
-u 'JD'  
-t source  
-t target  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

### ***Listing associations***

Enter the following command to list all associations for John S. Day:

```
eimadmin  
-lA  
-i 'John S. Day'  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

## Removing a user

To completely erase a person's identity from your EIM domain, remove the identifier.

If you only need to reflect the deletion of a user ID from a registry, simply remove the corresponding EIM associations.

### **Removing associations**

Enter the following command to remove the source and target associations for user ID JD in registry RACF Pok1:

```
eimadmin
-pA
-i 'John S. Day'
-r 'RACF Pok1'
-u 'JD'
-t source
-t target
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

### **Removing an identifier**

Enter the following command to remove an identifier and its associations, including identifier aliases:

```
eimadmin
-pI
-i 'John S. Day'
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'
-w secret
```

## Changing access authority

A user is permitted to perform EIM administrative or lookup operations based on the authority groups containing the user's LDAP DN. The user's DN is determined by the credentials authenticated when connecting to LDAP.

Suppose that a user has registry administrator authority over a specific registry, and your task is to switch the user's authority to a different registry. You can accomplish this task in two steps:

1. Add the user to the new registry administrator group.
2. Remove the user from the prior group.

### **Adding access authorities**

Enter the following command to add user DN `cn=Reggie King,ou=dept20,o=IBM,c=US` to the registry administration group for RACF Pok1:

```
eimadmin
-aC
-q 'cn=Reggie King,ou=dept20,o=IBM,c=US'
-f DN
-c registry
-r 'RACF Pok1'
```

```
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

### ***Listing access authorities***

Enter the following command to list all EIM access authorities for the user:

```
eimadmin  
-lC  
-q 'cn=Reggie King,ou=dept20,o=IBM,c=US'  
-f DN  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

### ***Removing access authorities***

Enter the following command to remove the user from the prior registry administration group for RACF Test Pok1:

```
eimadmin  
-pC  
-q 'cn=Reggie King,ou=dept20,o=IBM,c=US'  
-f DN  
-c registry  
-r 'RACF Test Pok1'  
-d 'ibm-eimDomainName=MyDomain,o=IBM,c=US'  
-h ldap://some.ldap.host  
-b 'cn=eimadministrator,ou=dept20,o=IBM,c=US'  
-w secret
```

Figure 4-28 on page 223 shows RACF support for EIM.

## 4.23 RACF support for EIM

- Security administrator has ability to
  - Define default EIM domain by system or by server
  - Define default LDAP bind information for the EIM domain
- Enhanced commands and profiles
  - ADDUSER, ALTUSER, LISTUSER
  - RDEFINE, RALTER, LISTUSER
  - IRR.PROXY.DEFAULTS FACILITY class profile
  - IRR.EIM.DEFAULTS LDAPBIND class profile

Figure 4-28 RACF support for EIM

### Using RACF for EIM domain access

The RACF administrator can use RACF commands to do the following functions:

- ▶ Add an EIM domain name and bind information for system-wide use
- ▶ Add an EIM domain name and bind information for use by a server
- ▶ Add an EIM domain name and bind information for use by an administrative user
- ▶ Assign a name to the local RACF registry for use by a lookup application

**Tip:** Issuing these commands is optional. However, setting up your system this way can eliminate the need for individual applications to handle EIM domain and bind information.

The default domain and bind information can be specified in one of three places:

1. The user ID that the application runs under has the name of an LDAPBIND class profile in its USER profile
2. The IRR.EIM.DEFAULTS profile in the LDAPBIND class
3. The IRR.PROXY.DEFAULTS profile in the FACILITY class

These RACF profiles can be set up in such a way as to control the access that the application has to the EIM domain:

- ▶ New connections with an EIM domain can be enabled or disabled by using keywords on the **RDEFINE** or **RALTER** commands.
- ▶ Bind credentials can be specific to the server or administrator who uses them.

The EIM APIs try to retrieve the information from a profile if the application does not explicitly supply the information to the EIM APIs using parameters. Applications or other services that use EIM can instruct their callers to define a profile in the LDAPBIND class profile.

Storing LDAP bind information in a profile is shown in Figure 4-29 on page 224.

## 4.24 Storing LDAP binding information in a profile

If ...	Then ...
The EIM domain is in the default system LDAP directory ...	Set up the IRR.PROXY.DEFAULTS profile in the FACILITY class. (This is the simplest way to set up a profile.)
A server needs to reference an EIM domain that is not in the system default LDAP directory ... (This could be because the IRR.PROXY.DEFAULTS profile has different bind information than the application using the EIM domain requires.)	Set up a profile in the LDAPBIND class.  Add the name of the LDAPBIND class profile to the user profile used by the application.

Figure 4-29 Storing LDAP bind information in a profile

Before you begin, use the decision table shown in Figure 4-29 to determine which profile to use.

### Adding EIM domain and bind information for servers or administrative users

To create a profile for LDAP binding information, follow these steps:

1. If you are creating a profile in the LDAPBIND class, define the domain in the LDAPBIND class. Enter the following command:

```
RDEFINE LDAPBIND racfProfileName EIM(DOMAINDN(domainDN))  
PROXY(LDAPHOST(ldapHost) + BINDDN(bindDN) BINDPW(bindPasswd))
```

2. To update the user profile:

```
ADDUSER ASERVER EIM(LDAPPROF(racfProfileName))
```

### Adding a system default using the IRR.EIM.DEFAULTS profile

If you are using the IRR.PROXY.DEFAULTS profile in the FACILITY class, enter:

```
RDEFINE LDAPBIND IRR.EIM.DEFAULTS PROXY(LDAPHOST(ldapHost) BINDDN(bindDN) +  
BINDPW(bindPasswd)) EIM(DOMAINDN(domainDN))
```

### Adding a system default using the IRR.PROXY.DEFAULTS profile

If no LDAPBIND class profile is associated with the caller's user profile, the EIM services look for the EIM domain's LDAP URL and binding information in the IRR.EIM.DEFAULTS profile in the LDAPBIND class followed by the IRR.PROXY.DEFAULTS profile in the FACILITY class.

For example, the following command sets up the binding information in the IRR.PROXY.DEFAULTS profile in the FACILITY class:

```
RDEFINE FACILITY IRR.PROXY.DEFAULTS PROXY(LDAPHOST(LDAP://SOME.BIG.HOST:389) +  
BINDDN('?cn=Joes Admin,o=IBM,c=US?') BINDPW(secret)) +  
EIM(DOMAINDN('?ibm-eimDomainName=Joes Domain,o=IBM,c=US?'))
```

In this case, the domain's LDAP URL is:

```
LDAP://SOME.BIG.HOST:389/ibm-eimDomainName=Joes Domain,o=IBM,c=US
```

Setting up a registry name for your local RACF registry is shown in Figure 4-30 on page 225.

## 4.25 Setting up a registry name for your local RACF registry

Information needed	Where to get it	Value
<i>registryName</i> —  The name of the RACF registry.  <b>Example:</b> Registry on POK System	EIM administrator	

Figure 4-30 Setting up a registry name for your local RACF registry

Many of the EIM APIs require the name of a registry. For example, if you are adding a registry to an EIM domain, know the name of the new registry. However, you can use the lookup APIs (such as `eimGetTargetFromSource`, `eimGetIdentifierFromSource`, and `eimGetAssociatedIdentifiers`) to convert:

1. A user ID to its equivalent RACF user ID
2. A local RACF user ID to an enterprise identifier

For such applications, you can eliminate the requirement for providing the RACF registry name or its alias on the local system. You do this by giving a name to the local RACF registry.

### Steps for setting up lookups that do not need a registry name

Before you begin, you need to know the registry name.

To set up EIM so that you do not need a registry name on every lookup, follow the instructions in this section. To define the local registry, enter the following RACF command in which *registryName* is the name of the local registry:

```
RDEFINE FACILITY IRR.PROXY.DEFAULTS EIM(LOCALREGISTRY(registryName))
```

**Note:** EIM does not look for the registry name in an LDAPBIND class profile.

You can also configure the system with a Kerberos registry name and an X.509 registry name. Issue the following commands to define default Kerberos and X.509 registries for the configured EIM domain:

```
RALTER FACILITY IRR.PROXY.DEFAULTS EIM(KERBREGISTRY(registry name) +  
X509REGISTRY(registry name))
```

This access can be removed with the following command:

```
RALTER FACILITY IRR.PROXY.DEFAULTS EIM(NOKERBREGISTRY NOX509REGISTRY)
```

**Note:** You need to define these registry names in the configured EIM domain.

### Disabling use of an EIM domain

You might need to temporarily disable use of a RACF profile with a configured EIM domain or a system-wide default EIM domain. You might want to do this if the EIM information in a domain has been compromised or a security administrator wants to stop the system or server from establishing new connections with the EIM domain. You can use RACF commands to disable a domain without deleting EIM information from the RACF profiles. When an EIM

domain is disabled through a RACF profile, existing connections to the domain complete their work. However, if an EIM service is trying to establish a connection with such a domain, the EIM service does not continue to look for an enabled domain.

If you want to disable a server (rather than a system) from using a configured EIM domain, enter the following command:

```
RALTER LDAPBIND ldapbind_profile EIM(OPTIONS(DISABLE))
```

This command applies only to a server that has an **ldapbind** class profile specified for its user ID.

**Tip:** To disable a system-wide default EIM domain (rather than a server) that default profiles use, enter one of the following commands:

```
RALTER FACILITY IRR.PROXY.DEFAULTS EIM(OPTIONS(DISABLE))  
RALTER LDAPBIND IRR.EIM.DEFAULTS EIM(OPTIONS(DISABLE))
```

## Using output from the RACF database unload utility and eimadmin to prime your EIM domain with information

You can start to put EIM information (identifiers, RACF user IDs, and associations) into your EIM domain by using output from DBUNLOAD and eimadmin.

For large installations, priming the EIM domain with identifiers and associations can involve a lot of work. To make the task of getting started with EIM easier, the **eimadmin** utility accepts as input a file containing a list of identifiers and associations.

The section explores the steps for setting up an EIM domain based on user information contained in a RACF database. The initial assumptions are that the EIM domain, World Wide Domain, has been created and an SAF system registry, SAF user IDs, is defined in the domain. The LDAP host name for the domain is `ldap://some.big.host`. The EIM administrator uses the bind distinguished name of `cn=EIM Admin,o=My Company,c=US` and the password is `secret`. The EIM administrator bind distinguished name has been given EIM administrator authority and can perform all of the steps that we list here.

A user with other types of EIM authority, such as the following types of authority, can perform a subset of the following steps:

- ▶ EIM identifier administrator authority only works with identifiers and source and target associations.
- ▶ EIM registries administrator authority only works with target associations.
- ▶ EIM registry-specific administrator authority for the SAF registry only works with target associations in the SAF registry.

To set up an EIM domain based on user information contained in a RACF database, follow these steps:

1. Request from your RACF security administrator a file containing a copy of the user profiles in the RACF database. The RACF security administrator can:
  - a. Run the database unload utility (IRRDBU00) to create the sequential file.
  - b. Run the file through a sort program, such as DFSORT or DFSORT ICETOOL to extract just the user profiles and wanted fields. The User Basic Data Record (0200) contains the user ID and the programmer name. In this example, the programmer name is used for the EIM identifier.



The DFSORT ICETOOL Report format has a 1 - 4 character name (for example, EIM). It contains the ICETOOL statements that control report format and record summary information, such as SORT, COPY, DISPLAY, and OCCURS statements. Example 4-33 shows a report format that can be used to extract RACF user IDs and the programmer names that are associated with the user IDs.

*Example 4-33 A sample report format*

---

```
*****
* Name: EIM
*
*
* Find all user IDs in the RACF database and their name
*****
COPY FROM(DBUDATA) TO(TEMPO001) USING(RACF)
OCCURS FROM(TEMPO001) LIST(PRINT) -
  TITLE('user IDs and Names') -
  ON(10,8,CH) HEADER('USER ID') -
  ON(79,20,CH) HEADER('Name')
```

---

The record selection criteria is as follows:

- The name of the member containing the record selection criteria is the report member name followed by CNTL (such as EIMCNTL).
- Record selection is performed using DFSORT control statements, such as **SORT** and **INCLUDE**.
- The **SORT** command is used to select and sort records.
- The **INCLUDE** command is used to specify conditions required for records to appear in the report.

2. When you receive the report from the security administrator, move it to a file in the HFS.
3. Add a **eimadmin** utility `?label line?` to the file containing user profiles. You can use any one of the editors available from the OMVS shell (such as OEDIT).
4. Add identifiers and list the results using the **eimadmin** shell command:

```
eimadmin
-aI
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

5. To list the identifiers that you added, issue the following command:

```
eimadmin
-lI
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

6. Create source and target associations between the identifiers and the user IDs in RACF. Because the file `racfUsers.txt` contains a label line that identifies user IDs as well as unique identifier names, it can be used to create associations:

```
eimadmin
-aA
-t source
```

```
-t target
-r"SAF user IDs"
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US" -w secret <racfUsers.txt
```

7. To list the associations that you added, issue the following command:

```
eimadmin
-lA
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

8. The following **eimadmin** commands can be used to give EIM Mapping Operations authority to each of the users (identified in the file `racfUsersDNs.txt`):

```
eimadmin
-aC
-c MAPPING
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-f DN
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsersDNs.txt
```

9. To list the accesses that have been granted, issue the following command:

```
eimadmin
-lC
-c MAPPING
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsersDNs.txt
```

**Tip:** At a minimum, a user who is looking for a mapping in the EIM domain needs to have EIM mapping operations authority. In most cases, the application has one set of credentials for connect to an EIM domain, and those credentials are shared by all users. However, if individual access is needed, a bind distinguished name needs to be defined for each of the users and given EIM mapping operations authority.

Open Cryptographic Enhanced Plug-ins are shown in Figure 4-31 on page 229.

## 4.26 Introduction to Open Cryptographic Enhanced Plug-ins

- what do we mean by OCEP
- what is a plug-in
- how does it work with a Security Server

Figure 4-31 Open Cryptographic Enhanced Plug-ins

### What we mean by OCEP

The Open Cryptographic Enhanced Plug-in (OCEP) is a feature supplied with Integrated Security Services, which provides some programs that operate within the Open Cryptographic Services Facility (OCSF) framework and provides some valuable services that can be used by the customer. It could be said that these modules are prebuilt examples to be used at specific locations within the OCSF framework.

### What a plug-in is

The supplied modules are called plug-ins because they are logically slotted into specific locations in the OCSF framework. OCEP supplies two service provider modules, which are called plug-ins. They are used within the OCSF Framework. In other words, they are actual implementations using the OCSF Framework.

Within this framework, these service provider modules are used within these two areas:

Trust Policy (TP module) is deployed to verify trust in a certificate, so the most crucial action by a TP module would be “Is this certificate trusted for this action”.

Data Storage Library (DL module) is a data storage library module (DL) that is involved in handling certificates and CRLs. It covers:

- ▶ Secure storage
- ▶ Persistent storage
- ▶ Retrieval: recovery of certificates and CRLs

An OCSF framework showing the location of OCEP plug-ins is shown in Figure 4-32 on page 230.

## 4.27 How OCEP works with a security server

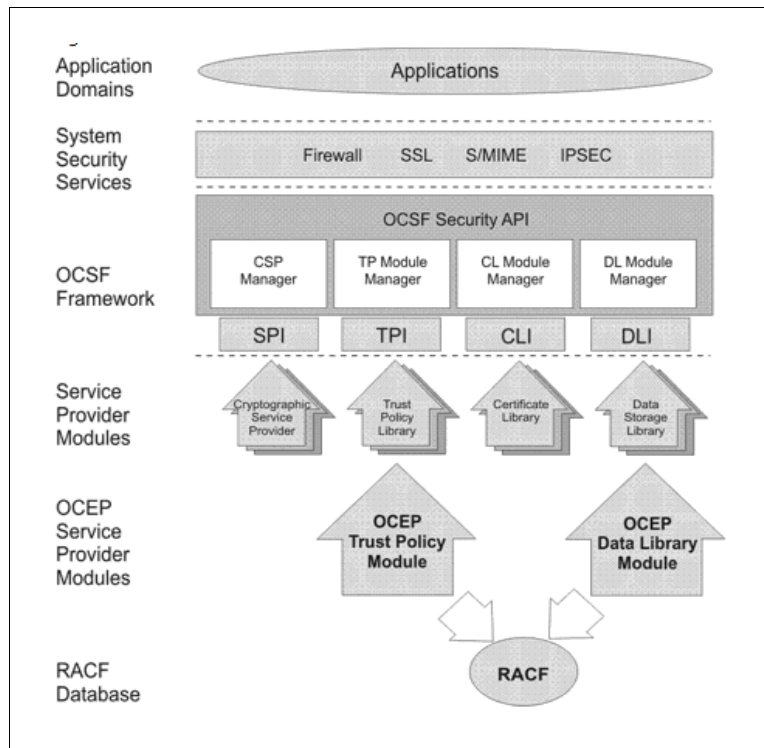


Figure 4-32 OCSF framework showing location of OCEP plug-ins

In general terms, the presence of these modules allows an application to use the z/OS Security Server (RACF) to provide security functions for digital certificates. Within this framework, these two service provider modules will also work with Certificate Library (CL modules) and Cryptographic Service Provider (CSP modules).

Remember that the OCSF Framework will handle interactions between the modules and the applications that use them. This is best described in Figure 4-32, where the two supplied modules are the OCEP Service Provider modules.

See *z/OS Open Cryptographic Services Facility Application Programming*, SC14-7513, to learn how you can build and use modules within the OCSF framework.



# Cryptographic Services

This chapter examines the Cryptographic Services that are available on System zEC12 servers.

## 5.1 Introduction to cryptography



Figure 5-1 Introduction to cryptography

### Introduction to cryptography

The word *cryptography* literally means secret writing. Throughout history, information has been an asset that provides the owner a competitive advantage.

Failure to adequately protect information has had significant consequences for countries. Today, if an enterprise does not exercise due care in protecting sensitive information about others, it risks losing its competitive advantage and market share through industrial espionage or losses due to lawsuits.

Confidentiality important. In addition, the integrity (the assurance of validity) of information is critical to business success around the world. Commercial enterprises send contracts, private documents, money orders, and other legal documents across communication networks, all of which must arrive with the same content with which they were dispatched. Before the electronic age, paper, signatures, and seals were used to guarantee the integrity of a document. With electronic communication, another mechanism is required.

Cryptography is the only known practical method of protecting information that is transmitted electronically through communication networks. It can also be an economical way to protect stored information. As computing systems become increasingly exposed through increased computer literacy and reliance on distributed computing, the pervasiveness of cryptography will increase as industry seeks ways to protect their information assets.

Cryptographic capabilities are shown in Figure 5-2 on page 233.

## 5.2 Cryptographic capabilities

- Cryptographic capabilities:
  - Data confidentiality
  - Data integrity
  - Authentication and Identification
  - Electronic signature

*Figure 5-2 Cryptographic capabilities*

### **Cryptographic capabilities**

The use of cryptography provides many data-handling capabilities, such as data confidentiality, data integrity, authentication, and electronic signatures.

#### **Data confidentiality**

Traditionally, cryptography is a data scrambling method used to conceal the information content of a message. When a message is encrypted, the input plain text (unencrypted text) is transformed by an algorithm into enciphered text that hides the meaning of the message. This process involves a secret key that is used to encrypt and (later) decrypt the data. Without this secret key, the encrypted data is meaningless. To conceal a message without using cryptography, a secure physical communication line is required. With cryptography, only the secret data encryption key has to be transmitted by a secure method. The encrypted text can be sent using any public mechanism.

#### **Data integrity**

Although cryptography is best known for its ability to protect the confidentiality of data, it is also used to protect the integrity of data. For example, a cryptographic checksum, such as a message authentication code (MAC), can be calculated on arbitrary user-supplied text. The text and MAC are then sent to the receiver. The receiver of the message can verify the MAC appended to a message by recalculating the MAC for the message using the appropriate secret key and verifying that it matches the received MAC exactly.

#### **Authentication and identification**

Another use of cryptography is in personal identification, where the user knows a secret that can serve to authenticate his or her identity. For example, the user of an automatic teller machine (ATM) enters a magnetic stripe card to identify the account and the corresponding correct PIN to authenticate the user. An unauthorized person acquiring the card and attempting to use it is reduced to guessing the correct PIN. Because a PIN is typically four digits and because a user typically gets only three attempts to enter the correct PIN, this is very unlikely to happen.

#### **Electronic signature**

In normal business, a legal transaction is completed by a verifiable authorized signature (just sign on the dotted line). An analogous process is required by new electronic applications, such as Electronic Data Interchange (EDI). A digital signature is a means of achieving this by using cryptographic mechanisms. It assures the recipient that the message is authentic and

that only the owner of the key could have produced the digital signature. A digital signature, such as the Rivest-Shamir-Adleman (RSA) algorithm, is well-suited for message non-repudiation. Non-repudiation is the ability of a party to sign a message, such that he or she is unable to later deny having signed the message.

Figure 5-3 shows symmetric and asymmetric encryption algorithms.

## 5.3 Symmetric and asymmetric encryption algorithms

- Benefits of asymmetric-key encryption compared to symmetric-key encryption:
  - Using the public key, anyone can create an encrypted message which only the holder of the private key can decrypt.
  - Using the private key, an encrypted message can be created which could have been created only by the holder of the private key.
- Disadvantage of asymmetric-key encryption compared to symmetric-key encryption:
  - Much more computing power is required to encrypt and decrypt (as compared to symmetric-key encryption techniques)

*Figure 5-3 Symmetric and asymmetric encryption algorithms*

### **Symmetric and asymmetric encryption algorithms**

Today, two distinct classes of encryption algorithms are in use:

- ▶ Symmetric encryption algorithms
- ▶ Asymmetric encryption algorithms

Their fundamental difference is in how keys are used with these encryption methods.

We discuss these algorithms in the next sections (5.4, “Symmetric encryption algorithms” on page 235, and 5.5, “Asymmetric encryption algorithms” on page 236).

Figure 5-4 on page 235 shows symmetric encryption algorithms.



## 5.4 Symmetric encryption algorithms

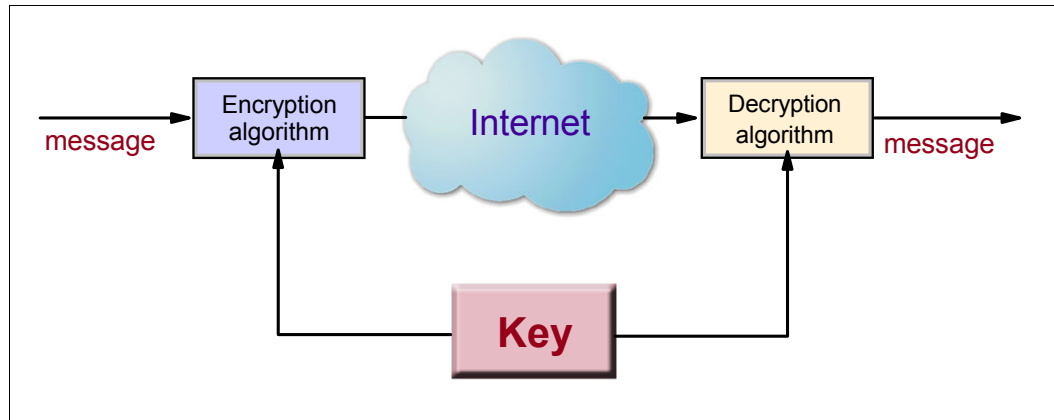


Figure 5-4 Symmetric encryption algorithms

### Symmetric encryption algorithms

An encryption algorithm is called *symmetric* when the same key that is used to encrypt the data is also used to decrypt the data and recover the plain text (see Figure 5-4). The cipher and decipher processes are usually mathematically complex non-linear permutations.

Most symmetric ciphers are block ciphers. They operate on a fixed number of characters at a time, usually 8 bytes. Following are some frequently used algorithms:

- ▶ **Data Encryption Standard (DES):** Developed in the 1970s by IBM scientists, DES uses an 8-byte key; however, one bit in each byte is used as a parity bit; so, the key length is 56 bits. Stronger versions called Triple DES, which uses three operations in sequence, have been developed:
  - 2-key Triple DES encrypts with key 1, decrypts with key 2, and encrypts again with key 1. The effective key length is 112 bits.
  - 3-key Triple DES encrypts with key 1, decrypts with key 2, and encrypts again with key 3. The effective key length is 168 bits.
- ▶ **Advanced Encryption Standard (AES):** Sometimes known as *Rijndael*, AES is a block cipher adopted as an encryption standard by the US government. It is considered the successor to DES and TDES and is expected to be used worldwide. AES uses a larger block size than DES and TDES do. While DES uses a block size of 8 bytes (64 bits), AES uses a block size of 16 bytes (128 bits) along with the capability of using longer keys than DES or TDES. This block size should be acceptable for messages of up to 256 exabytes of data, and the bigger length of the keys delays for quite a few years the possibility of finding the key value using brute force.
- ▶ **Commercial Data Masking Facility (CDMF):** A version of the DES algorithm that is used for export from the US and uses 56-bit keys; however, 16 bits of the key are known. So, the effective key length is 40 bits.
- ▶ **RC2:** Developed by Ron Rivest for RSA Data Security, Inc., RC2 is a block cipher with variable key length operating on 8-byte blocks. Key lengths of 40 bits, 64 bits, and 128 bits are used.
- ▶ **RC4:** Developed by Ron Rivest for RSA Data Security, Inc., RC4 is a stream cipher with variable key length. Stream ciphers operate on each byte, not on blocks of data. Key lengths of 40 bits, 64 bits, and 128 bits are used.

**Note:** Both RC2 and RC4 are proprietary confidential algorithms that have never been published. They have been examined by a selected number of scientists working under non-disclosure agreements.

With these ciphers, it can be assumed that a brute-force attack is the only means of breaking the cipher; therefore, the work factor depends on the length of the key. If the key length is  $n$  bits, the work factor is proportional to  $2^{(n-1)}$ .

Figure 5-5 shows asymmetric encryption algorithms.

## 5.5 Asymmetric encryption algorithms

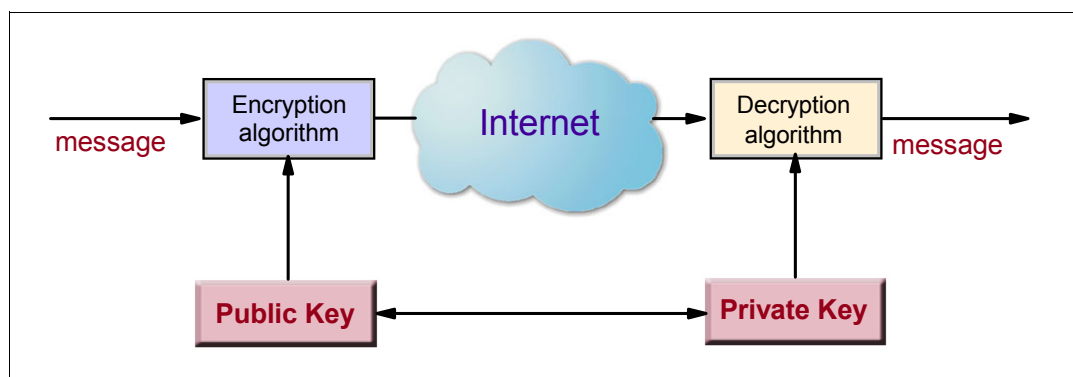


Figure 5-5 Asymmetric encryption algorithms

### Asymmetric encryption algorithms

An encryption algorithm is called *asymmetric* when the key that is used to encrypt the data cannot be used to decrypt the data. A different key is needed to recover the plain text (see Figure 5-5). This key pair is called a *public key* and a *private key*. If the public key is used to encrypt the data, the private key must be used to recover the plain text. If data is encrypted with the private key, it can only be decrypted with the public key.

Asymmetric encryption algorithms, commonly called *Public Key Cryptosystems* (PKCS), are based on mathematical algorithms. The basic idea is to find a mathematical problem that is very hard to solve. Only one algorithm, RSA, is in widespread use today. However, some companies have begun to implement public-key cryptosystems based on so-called elliptic curve algorithms. The following list provides a brief overview of asymmetric algorithms:

- ▶ **RSA:** RSA was invented in 1977 by Rivest, Shamir, and Adleman (who formed RSA Data Security, Inc.). The idea behind RSA is that integer factorization of very large numbers is extremely hard to do. Key lengths of public and private keys are typically 512 bits, 1024 bits, or 2048 bits.
- ▶ **Elliptic Curve:** Public-key cryptosystems based on elliptic curves use a variation of the mathematical problem to find discrete logarithms. It has been stated that an elliptic curve cryptosystem implemented over a 160-bit field has roughly the same resistance to attack as RSA with a 1024-bit key length.

Elliptic curve cryptosystems are said to have performance advantages over RSA in decryption and signing.

While the possible differences in performance between the asymmetric algorithms are somewhere in the range of a factor of 10, the performance differential between symmetric and asymmetric cryptosystems is far more dramatic. It takes about 1000 times longer to encrypt the same data with RSA as it does with DES, and implementing both algorithms in hardware does not change the odds in favor of RSA.

Cryptosystems for data privacy are shown in Figure 5-6.

## 5.6 Use of cryptosystems: Data privacy

- Cryptosystems for data privacy
  - Key management in a closed environment
  - Distributed computing environment (DCE)
  - Public key infrastructure

Figure 5-6 Uses of cryptosystems

### Uses of cryptosystems

*Cryptosystems*, both symmetric and asymmetric, are used for data privacy, data integrity, and digital signatures.

### Cryptosystems for data privacy

Encrypting and decrypting large amounts of data with asymmetric cryptosystems is expensive (regarding time and resources). Therefore, symmetric algorithms, such as AES, DES, RC2, or RC4, are used for bulk data encryption. The disadvantage of symmetric algorithms, however, is that both partners (the party that encrypts the data and the party that decrypts the data) must be in possession of the same key. Key management or safe distribution of keys in insecure networks is a problem with symmetric cryptosystems, even more so because data encryption keys need to be changed frequently in order to make an adversary's task more difficult and limit the potential damage if a key is compromised.

Different solutions exist in different environments, and we list a few of these solutions in the following sections.

#### **Key management in a closed environment**

In high-security environments using cryptographic hardware that is installed and managed by a centralized security facility, Master Keys and key-exchange keys can be installed centrally, and the hardware facility can be delivered to the users with the necessary keys installed.

If tamper-resistant hardware is used, this solution can fulfill the highest security requirements and can also be made secure against insider attacks. The amount of administrative effort and cost, however, would be prohibitive for many environments.

#### **Distributed Computing Environment**

Distributed Computing Environment (DCE) is designed to provide secure client/server computing in insecure networks. It uses DES, a symmetric cryptosystem.

Users (principals) are authenticated by a central authentication server (the DCE Security Server) using the Kerberos V.5 third-party authentication method. All client and server

principals must be defined in the registry (the authentication server's database). Client users have a password that they must remember, and servers have a key that is normally stored in a keyfile on the server's computer. The passwords and server keys are stored in the registry as the principals' Master Keys.

During authentication, the security server can send information to the client encrypted under the client's Master Key (password). A client who wants to communicate with an application server needs a ticket for this application server from the security server. A ticket is a collection of information about the client, encrypted by the security server with the Master Key of the application server. The client cannot read or modify the ticket, which can be compared to a sealed envelope that the client can forward to the server as a method for identify but which the user cannot open, read, or modify.

The security server creates a random session key that the client and the application server can use to encrypt the data that they send to each other. This session key is included in the ticket and is also sent to the client encrypted under the client's Master Key.

The authentication and key management method used by DCE can create a highly secure client/server environment. If all security features provided by DCE are used, a network can be made impenetrable even to sophisticated intruders. A hacker would need a computer that is defined in the registry with a valid Master Key to even be able to attempt to log in and make a guess at a principal's password.

The use of symmetric encryption causes the overhead for the security functions, although too large to be neglected to be tolerable.

The downside is that all clients need to be defined and administered in the registry. This is adequate for client/server computing within an enterprise but does not scale well into a user population made up of large numbers of suppliers and customers on the Internet.

### ***Public key infrastructure***

Public key cryptosystems can be used to transmit the DES, RC2, or RC4 keys used to encrypt data to the recipient. Data that has been encrypted with the public key of the recipient can only be decrypted using the recipient's private key. If someone makes the public key publicly known, everybody can send that person's encrypted data by using the following procedure:

1. Create a random DES, RC2, or RC4 key to encrypt the data.
2. Encrypt the data using this key.
3. Encrypt the key with the RSA PKCS using the recipient's public key.
4. Append the encrypted key before or after the encrypted data.
5. Send to recipient.

The recipient uses the private key to decrypt the DES, RC2, or RC4 key and uses this key to decrypt the data and recover the plain text. This method works very well and has reasonable performance because RSA is used to encrypt or decrypt only small amounts of data. The length of symmetric keys is typically 8 - 32 bytes.

The problem with this method arises from the question: How can someone publish a public key in a secure manner? If I send you my public key, pretending it is the public key of someone else (for example, Jack Jones), and trick you into believing me, you will then send encrypted data to the person who you believe is Jack Jones, and I can decrypt that data. This situation is one where digital certificates and a public key infrastructure (PKI), a hierarchy of authorities that issue certificates and attest to their authenticity, can help.

Figure 5-7 on page 239 shows cryptosystems for data integrity.

## 5.7 Use of cryptosystems: Data integrity

- Cryptosystems for data integrity
  - Message authentication codes
  - Message digest algorithms

Figure 5-7 Data integrity

### Cryptosystems for data integrity

*Data integrity* is the ability to assert that the data that is received over a communication link is identical to the data that is sent. Data integrity in an insecure network requires the use of cryptographic algorithms, but it does not imply that only the receiver can read the data, as is the case with data privacy. Data can be compromised not only by an attacker, but it can also be damaged by transmission errors (although these are normally handled by the transmission protocols).

#### **Message authentication codes**

Symmetric cryptographic algorithms, such as DES, can be used for data integrity. Using a variation of the DES algorithm and a secret key, an 8-byte message authentication code (MAC) is created from the data. The MAC is sent with the message. The receiver performs the same operation using the same key and compares the resulting MAC with the MAC that was sent with the data. If both match, the integrity of the data is assured.

MACs rely on the same secret key that is used by both the sender (to create the MAC) and the receiver (to verify the MAC). Since the MAC is derived from a secret key known only to the sender and receiver the MAC can be sent in the clear. An adversary sitting between the sender and the receiver (a so-called “person-in-the-middle” attack) can alter the message but cannot forge the MAC because the key to create the MAC is unknown. The mathematical principle behind using the MAC is that finding a message that fits a certain MAC is as difficult as breaking DES encryption.

A disadvantage to this method is that, as in symmetric cryptosystems, secret keys must be shared by sender and receiver. Furthermore, because the receiver has the key that is used in MAC creation, it is difficult to make it impossible for the receiver to forge a message and claim it was sent by the sender.

#### **Message digest algorithms**

Message digesting algorithms are a different approach to data integrity. These are algorithms that digest (condense) a block of data into a shorter string (usually 128 or 160 bits) called a Message Digest, Secure Hash, or Message Integrity Code (MIC).

The principles behind message digesting algorithms are:

- ▶ The message cannot be recovered from the message digest.
- ▶ It is hard to construct a block of data that has the same message digest as another given block.

Some common message-digesting algorithms are:

- ▶ **MD2:** This algorithm was developed by Ron Rivest of RSA Data Security, Inc. The algorithm is used mostly for PEM certificates. MD2 is fully described in RFC 1319. Because weaknesses have been discovered in MD2, its use is discouraged.

- ▶ **MD5:** This algorithm was developed in 1991 by Ron Rivest. The algorithm takes a message of arbitrary length as input and produces as output a 128-bit message digest of the input. The MD5 message digest algorithm is specified by RFC 1321, The MD5 Message-Digest Algorithm.
- ▶ **SHA-1:** This algorithm was developed by the US Government. The algorithm takes a message of arbitrary length as input and produces as output a 160-bit hash of the input. SHA-1 is fully described in standard FIPS 180-1.
- ▶ **SHA-2:** SHA-256 is an improved algorithm and generates a 32-byte hash value. SHA-256 is considered to generate message digest values that are less likely to yield collisions.
- ▶ **MDC-4:** The MDC-4 algorithm calculation is a one-way cryptographic function that is used to compute the hash pattern of a key part. MDC uses encryption only, and the default key is 5252 5252 5252 5252 2525 2525 2525 2525. It is used by the Trusted Key Entry (TKE).

The sender of a message (block of data) uses an algorithm (for example SHA-1) to create a message digest from the message. The message digest is sent together with the message. The receiver runs the same algorithm over the message and compares the resulting message digest to the one sent with the message. If both match, the message is unchanged.

The message digest cannot be sent in the clear. Because the algorithm is well known and no key is involved, a person-in-the-middle can forge the message and can also replace the message digest with that of the forged message, making it impossible for the receiver to detect the forgery. Depending on the application and the key management used, either symmetric cryptosystems or public-key cryptosystems can be used to encrypt the message digest.

Because a message digest is a relatively small amount of data, it is especially well-suited for public-key encryption.

## 5.8 Use of cryptosystems: Digital signatures



Figure 5-8 Digital signatures

### Digital signatures

*Digital signatures* are an extension of data integrity. While data integrity only ensures that the data received is identical to the data that is sent, digital signatures go a step further. They provide *non-repudiation*, which means that the sender of a message (or the signer of a document) cannot deny authorship (similar to signatures on paper).

The creator of a message or electronic document that is to be signed uses a message digesting algorithm, such as MD5 or SHA-1, to create a message digest from the data. The message digest and some information that identifies the sender are then encrypted with the sender's private key. This encrypted information is sent together with the data.

The receiver uses the sender's public key to decrypt the message digest and sender's identification. The receiver then uses the message digesting algorithm to compute the message digest from the data. If this message digest is identical to the one recovered after

decrypting the digital signature, the message is authentic, and the signature is recognized as valid.

With digital signatures, only public-key encryption can be used. If symmetric cryptosystems are used to encrypt the signature, it is very difficult to make sure that the receiver (having the key to decrypt the signature) could not misuse this key to forge a signature of the sender. The private key of the sender is not known to anyone else. So, nobody can forge the sender's signature.

The difference between encryption using public key cryptosystems and digital signatures includes:

- ▶ With encryption, the sender uses the receiver's public key to encrypt the data, and the receiver decrypts the data with a private key. Thus, everybody can send encrypted data to the receiver that only the receiver can decrypt.
- ▶ With digital signatures, the sender uses the private key to encrypt the signature, and the receiver decrypts the signature with the sender's public key. Thus, only the sender can encrypt the signature, but anyone who receives the signature can decrypt and verify it.

The tricky thing with digital signatures is the trustworthy distribution of public keys.

Figure 5-9 on page 242 shows IBM Common Cryptographic Architecture.

## 5.9 IBM Common Cryptographic Architecture

- CCA and extended services
  - Managing DES Cryptographic Keys
  - Protecting Data
  - Verifying Data Integrity and Authenticating Messages
  - Financial Services
  - Using Digital Signatures
  - Managing PKA Cryptographic Keys
  - Utilities
  - Trusted Key Entry Workstation Interfaces

Figure 5-9 IBM CCA

### **IBM Common Cryptographic Architecture**

The IBM Common Cryptographic Architecture (CCA), defines a set of cryptographic functions, external interfaces, and key management rules that pertain both to the DES-based symmetric algorithms and the public key algorithm (PKA) asymmetric algorithms. These provide a consistent, end-to-end cryptographic architecture across different platforms that conforms to American and International Standards.

Functions of the CCA define services for:

- ▶ Key management, which includes generation and exchange of keys securely across networks and between application programs. The exchanged key is encrypted securely using either DES or a PKA used in the context of symmetric key management.
- ▶ Data integrity, with the use of a message authentication code (MAC), Modification Detection Code (MDC), or digital signature.
- ▶ Data confidentiality, with the use of encryption and decryption capabilities accessible at all levels of a network protocol stack.
- ▶ Personal authentication, with PIN generation, verification, and translation.

CCA was introduced in October 1989 with the IBM Transaction Security System and the IBM Integrated Cryptographic Facility (IBM ICSF) with its supporting Integrated Cryptographic Services Facility/MVS (ICSF/MVS).

These products and their follow-ons conform to the IBM CCA application programming interface.

### **CCA key management functions**

Key management is essential to successful cryptography. Because the algorithm is usually public knowledge, the security of the data depends on the security of the key that is used to encipher the data. Enciphered data can be obtained by an adversary, but without access to the cryptographic key, the data remains secure.

Key management in the IBM CCA includes the following:

- ▶ **Master Key concept:** Each cryptographic system has a *Master Key* that is kept in the clear inside the cryptographic facility, which is a highly secured physical repository. Each



operational DES key is encrypted under the appropriate Master Key variant (see 5.14, “DES key management” on page 251), allowing an installation to protect many keys while providing physical protection for only one key.

- ▶ **PKA keys:** The concept of Master Key is also applied to PKA keys that are encrypted under the PKA Master Key.
- ▶ **Key separation:** Cryptographic keys should be used only for their intended function. For DES keys, the IBM CCA enforces key separation by using control vectors (CVs).

A control vector is a fixed pattern defined for each key type that the cryptographic facility exclusively ORs with the Master Key to produce a Master Key variant that is used to encrypt the key. Effectively, this produces a unique Master Key for each key type. The Master Key variants protect keys operating on the system; these are called *operational keys*.

The control vector concept also applies to the secure transportation of symmetric keys, where the transported key is encrypted under a variant of the key-encrypting-key. For example, when a key is stored with a file or sent to another system, the key is encrypted under a key-encrypting key.

## CCA API

The IBM CCA cryptographic API definition uses a *common key management approach* and contains a set of *consistent callable services*. (A callable service is a routine that receives control when an application program issues a CALL statement.)

Common key management ensures that all products that conform to the architecture allow users to share cryptographic keys in a consistent manner. The definition of key management provides methods for initializing keys on systems and networks, and also supports methods for the generation, distribution, exchange, and storage of keys.

Table 5-1 shows most of the categories of CCA callable services and some of the services in each category. The *service pseudonym* is the descriptive name for a service, while the *service name* is the formal name for the service and the name by which the service is called from a program.

Table 5-1 Some CCA callable services

Service pseudonym	Service name
<b>Managing DES cryptographic keys</b>	
Clear key import	CSNBCKI
Data key export	CSNBKX
Data key import	CSNBKIM
Key export	CSNBKEX
Key generate	CSNBKGN
Key import	CSNBKIM
Random number generate	CSNBRNG
Symmetric key export	CSNDSYX
Symmetric key generate	CSNDSYG
Symmetric key import	CSNDSYI
<b>Protecting data</b>	
Decipher	CSNBDEC

<b>Service pseudonym</b>	<b>Service name</b>
Encipher	CSNBENC
Symmetric key decipher	CSNBSYD
Symmetric key encipher	CSNBSYE
<b>Verifying data integrity/authenticity</b>	
MAC generate	CSNBMGN
MAC verify	CSNBMVR
One-way hash generate	CSNBOWH
<b>Financial services</b>	
Clear PIN encrypt	CSNBCPE
Clear PIN generate	CSNBPGN
Encrypted PIN generate	CSNBEPG
Encrypted PIN verify	CSNBPVR
<b>Using digital signatures</b>	
Digital signature generate	CSNDDSG
Digital signature verify	CSNDDSV
<b>Managing PKA cryptographic keys</b>	
PKA key generate	CSNDPKG
PKA key import	CSNDPKI
PKA key token build	CSNDPKB
PKA public key extract	CSNDPKX

- Figure 5-10 on page 245 shows a cryptographic overview of IBM System zEC12.

## 5.10 IBM System zEC12: Cryptographic overview

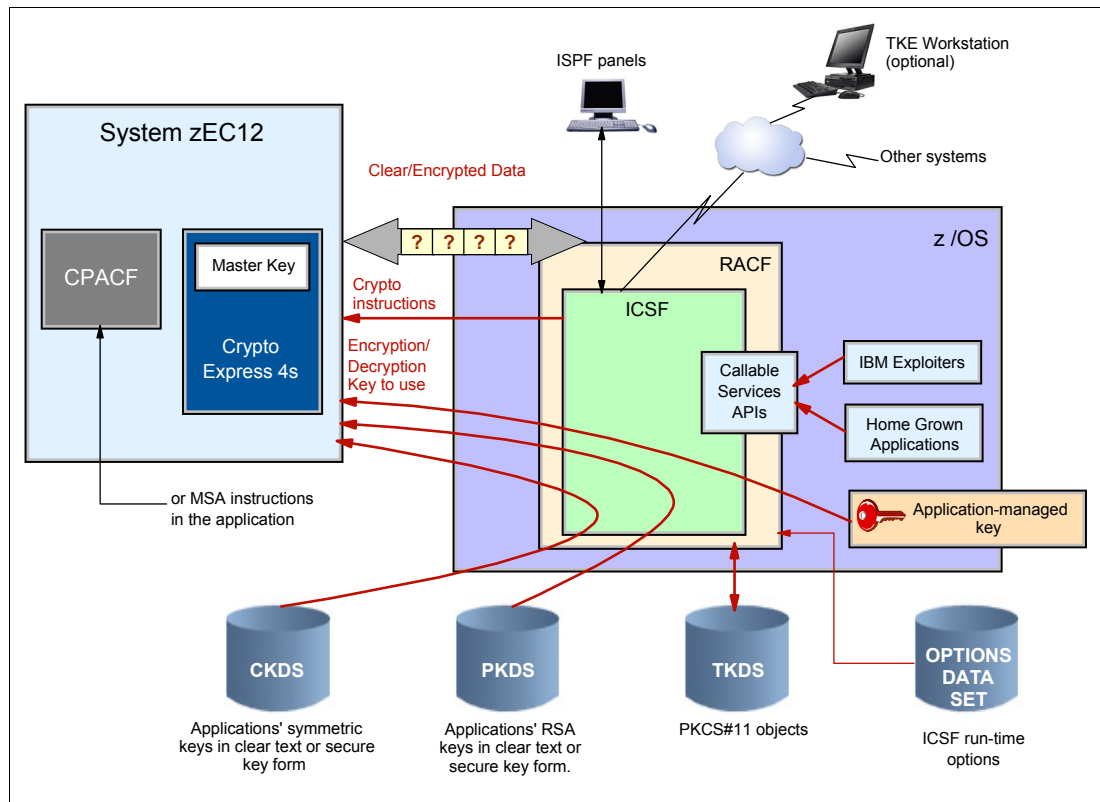


Figure 5-10 IBM System zEC12: Cryptographic overview

### IBM System z zEC12: Cryptographic overview

Three types of cryptographic hardware features are available on System z zEC12.

- ▶ CP Assist for Cryptographic Function (CPACF).
- ▶ The Crypto Express4S feature configured as a coprocessor (CEX4C for CCA and CEX4P for PKCS#11) or as an accelerator (CEX4A). It is available on the IBM zEnterprise® EC12.
- ▶ Crypto Express3 feature configurable as a coprocessor (CEX3C) or as an accelerator (CEX3A).

The CPACF features are usable only when explicitly enabled through Feature Code 3863, except for the CPACF SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 functions, which are always enabled.

To fully use the zEC12 cryptographic features requires the Integrated Cryptographic Service Facility (ICSF), which is the support program for the cryptographic features CPACF, CEX4C, CEX4P, CEX4A, CEX3C, and CEX3A. ICSF is integrated into z/OS.

Additionally, the optional TKE workstation feature is part of a customized solution for using the Integrated Cryptographic Service Facility for z/OS licensed program to manage cryptographic keys of a System zEC12 that has CEX4C, CEX4P, or CEX3C features installed and intended for the use of DES and PKA with secure cryptographic keys.

The TKE workstation provides secure control of the CEX4C, CEX4P, and CEX3C features, including loading of master keys. The TKE workstation is required for management of the Crypto Express4S when defined as an CEX4P coprocessor.

Figure 5-10 on page 245 describes the overall hardware and software layout of the hardware cryptography in a System zEC12 and z/OS, as follows:

- ▶ The exploiters of the cryptographic services call the ICSF API. Some functions are performed by the ICSF software without invoking the cryptographic coprocessor; other functions result in ICSF going into routines containing the cryptographic instructions. The cryptographic instructions to drive CEX4C and CEX3C are IBM proprietary and are not disclosed; the cryptographic instructions to interface with CPACF are published in *z/Architecture Principles of Operation*, SA22-7832.
- ▶ These instructions are executed by a CPU engine and, if not addressing the CPACF functions, result in a work request being generated for a cryptographic coprocessor.

The cryptographic coprocessor is provided with the following:

- ▶ Data to encrypt or decrypt from the system memory.
- ▶ The key used to encrypt or decrypt provided by ICSF according to the exploiter's request.

**Note:** The encryption or decryption keys are themselves encrypted and, therefore, unusable when residing outside of the cryptographic coprocessor.

Physically, these keys can be stored in ICSF-managed VSAM data sets and pointed to by the application using the label they are stored under. The Cryptographic Key Data Set (CKDS) is used to store the symmetric keys in their encrypted form, and the Public Key Data Set (PKDS) is used to store the asymmetric keys. The application also has the capability of providing an encrypted encryption key or a clear encryption key directly in memory (that is to use *as is*) to the coprocessor.

For high-speed access to symmetric cryptographic keys, the keys in the CKDS are duplicated into an ICSF-owned data space.

Figure 5-11 on page 247 shows hardware implementation of CP Assist for Cryptographic Functions.

## 5.11 CP Assist for Cryptographic Functions

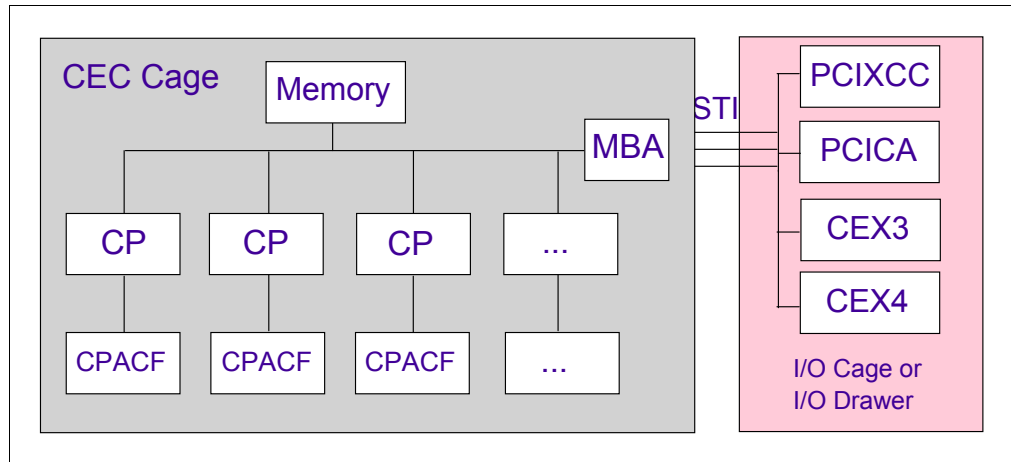


Figure 5-11 CPACF hardware implementation

### CPACF hardware implementation

CP Assist for Cryptographic Functions (CPACF) was first introduced on the z990 and z890. CPACF provides for hash functions, clear key encryption, and decryption functions and protected key functions. Each system central processor (CP) has an assist processor on the chip in support of cryptography.

CPACF operates with a specific set of machine instructions, the Message-Security Assist (MSA) instructions, which are problem state instructions and therefore available to all applications. Alternatively, these functions can also be called through the Integrated Cryptographic Service Facility (ICSF) component of z/OS by an ICSF-aware application. The MSA instructions are described in *z/Architecture Principles of Operation*, SA22-7832.

The MSA instructions are all executed synchronously regarding the CP instruction stream, contrary to the operations executed on the Crypto Express cards, which execute asynchronously. The CPACF operations are therefore quite fast and can be used to support a high volume of cryptographic requests. Because the CPACF instructions are available on every PU within a System zEC12 and because the CPACF operates with clear keys only, there is no notion of logical partition sharing or cryptographic domains with CPACF.

The CPACF provides the MSA instruction set on *every* CP of a zEC12 and zBC12 server.

MSA provides the following instructions:

<b>KMAC</b>	Compute Message Authentic Code
<b>KM</b>	Cipher Message
<b>KMC</b>	Cipher Message with Chaining
<b>KMF</b>	Cipher Message with CFB
<b>KMCTR</b>	Cipher Message with Counter
<b>KMO</b>	Cipher Message with OFB
<b>KIMD</b>	Compute Intermediate Message Digest
<b>KLMD</b>	Compute Last Message Digest
<b>PCKMO</b>	Provide Cryptographic Key Management Operation

Each of these instructions can perform several functions. Therefore, the MSA basic facility supplies a query function with each instruction so that the programmer can determine whether a given function is available on a given processor. If a programmer attempts to use a

function that is not available, his program will get a program interruption with interruption code 6 (specification exception). In z/OS, this is normally presented as an 0C6 abend.

On the zEC12 and zBC12, the MSA instruction set always includes the following functions:

- ▶ KIMD-SHA-1, KIMD-SHA-256, KIMD-SHA-512 and KIMD-GHASH
- ▶ KLMD-SHA-1, KLMD-SHA-256 and KLMD-SHA-512

Feature 3863 is required to enable the additional instructions.

Because the CPACF cryptographic functions are implemented in each CP, the potential throughput scales with the number of CPs in the server.

The hardware of the CPACF that performs encryption operations and SHA functions operates synchronous to the CP operations. The CP cannot perform any other instruction execution while a CPACF cryptographic operation is being executed. The CP internal code performs data fetches and stores resultant data while cryptographic operations are executed in the CPACF hardware on a unit basis as defined by the hardware.

Figure 5-12 shows the Crypto Express4S feature.

## 5.12 Crypto Express4S feature



Figure 5-12 Crypto Express4S feature

### Crypto Express4S feature

The Crypto Express4S feature (FC 0865) is an optional zEC12 exclusive feature. Each feature has one PCIe cryptographic adapter. The Crypto Express4S feature occupies one I/O slot in a zEC12 PCIe I/O drawer. This feature provides a secure programming and hardware environment on which crypto processes are run. Each cryptographic coprocessor includes a general-purpose processor, non-volatile storage, and specialized cryptographic electronics. The Crypto Express4S feature provides tamper-sensing and tamper-responding, high-performance cryptographic operations.

Each Crypto Express4S PCI Express adapter can be in one of these configurations:

- ▶ Secure IBM CCA coprocessor (CEX4C) for Federal Information Processing Standard (FIPS) 140-2 Level 4 certification. This configuration includes secure key functions. It is optionally programmable to deploy more functions and algorithms by using user programming function (UDX).
- ▶ Secure IBM Enterprise PKCS #11 (EP11) coprocessor (CEX4P) implements an industry standardized set of services that adhere to the PKCS #11 specification v2.20 and more recent amendments. It was designed for extended FIPS and Common Criteria evaluations to meet public sector requirements. This new cryptographic coprocessor mode introduced the PKCS #11 secure key function.

TKE workstation is required to support the administration of the Crypto Express4S when configured as EP11 mode.

- ▶ Accelerator (CEX4A) for acceleration of public key and private key cryptographic operations that are used with SSL/Transport Layer Security (TLS) processing.

These modes can be configured by using the Support Element, and the PCIe adapter must be configured offline to change the mode.

The Crypto Express4S uses the IBM 4765 PCIe Coprocessor. The Crypto Express4S feature does not have external ports and does not use fiber optic or other cables. It does not use channel-path identifiers (CHPIDs), but requires one slot in the PCIe I/O drawer and one physical channel ID (PCHID) for each PCIe cryptographic adapter. Removal of the feature or card *zeroizes* its content. The zEC12 supports a maximum of 16 Crypto Express4S features. Access to the PCIe cryptographic adapter is controlled through the setup in the image profiles on the SE.

Each zEC12 supports up to 16 Crypto Express4S features.

Figure 5-13 on page 250 shows the Crypto Express3 feature.

## 5.13 Crypto Express3 feature

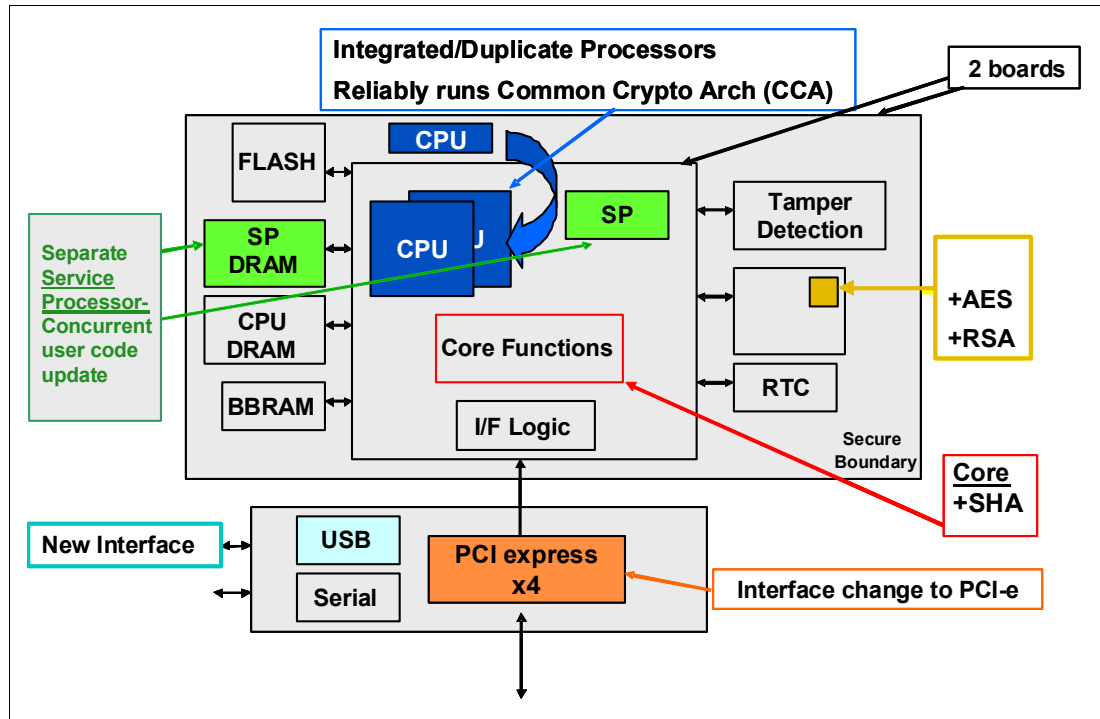


Figure 5-13 Crypto Express3 feature

### Crypto Express3 feature

The Crypto Express3 feature (FC 0864) is an optional feature, and is available only on a carry-forward basis when you upgrade from earlier generations to zEC12. Each feature has two PCIe cryptographic adapters. The Crypto Express3 feature occupies one I/O slot in an I/O cage or an I/O drawer.

Each Crypto Express3 PCI Express adapter can have one of these configurations:

- ▶ Secure coprocessor (CEX3C) for Federal Information Processing Standard (FIPS) 140-2 Level 4 certification. This configuration includes secure key functions, and is optionally programmable to deploy more functions and algorithms by using UDX.
- ▶ Accelerator (CEX3A) for acceleration of public key and private key cryptographic operations that are used with SSL/TLS processing.

These modes can be configured by using the Support Element. The PCIe adapter must be configured offline to change the mode.

The Crypto Express3 feature is designed to complement the functions of CPACF. This feature is tamper-sensing and tamper-responding. Unauthorized removal of the adapter or feature *zeroizes* its content. It provides dual processors that operate in parallel, supporting cryptographic operations with high reliability.

The CEX3 uses the 4765 PCIe Coprocessor. It holds a secured subsystem module, batteries for backup power, and a full-speed USB 2.0 host port available through a mini-A connector. On System z, these USB ports are not used. The securely encapsulated subsystem contains two 32-bit IBM PowerPC® 405D5 RISC processors that run in lockstep with cross-checking to



detect malfunctions. The subsystem also includes a separate service processor that is used to manage these items:

- ▶ Self-test and firmware updates
- ▶ RAM, flash memory, and battery-powered memory
- ▶ Cryptographic-quality random number generator
- ▶ AES, DES, TDES, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, and modular-exponentiation (for example, RSA, DSA) hardware
- ▶ Full-duplex direct memory access (DMA) communications

The Crypto Express3 feature does not have external ports, and does not use fiber optic or other cables. It does not use CHPIDs, but requires one slot in the I/O cage and one PCHID for each PCIe cryptographic adapter. Removal of the feature or card *zeroizes* the content.

The zEC12 supports a maximum of eight Crypto Express3 features on a carry-forward basis, offering a combination of up to 16 coprocessors and accelerators. Access to the PCIe cryptographic adapter is controlled through the setup in the image profiles on the Support Element (SE).

Each zEC12 supports up to eight Crypto Express3 features, which means a maximum of 16 PCIe cryptographic adapters.

## 5.14 DES key management

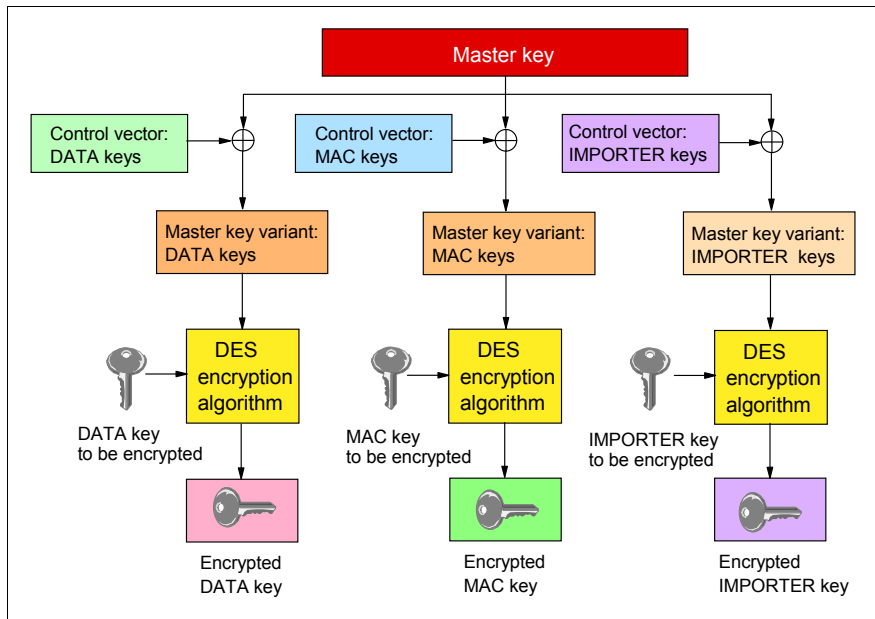


Figure 5-14 DES key management

### DES key management

Because the DES and TDES algorithms are controlled by keys, the security of protected data depends on the security of the cryptographic key. The CCA uses a master key to protect other keys. Keys are active on a system only when they are encrypted under a variant of the master key, so the master key protects all keys that are used on the system. A master key always remains in a secure area in the cryptographic hardware. In a z/OS environment, an ICSF

administrator initializes and changes master keys using the ICSF panels or a TKE workstation.

All other keys that are encrypted under a master key are stored outside the protected area of the cryptographic hardware; they cannot be attacked because the master key used to encrypt them is itself secure inside the tamper-protected cryptographic hardware and is zeroized if there is any attempted attack. This is an effective way to protect many keys while needing to provide physical security for only a master key.

When the cryptographic hardware is a CEX3C/CEX4C, the master key is called the *Symmetric-keys Master Key* (SYM-MK). In a z/OS environment, the SYM-MK is 128 bits (16 bytes) long.

### Cryptographic key separation

An important concept used in the CCA cryptographic API is cryptographic key separation. This concept provides for the creator of a cryptographic key (for example, using the Key Generate service) to declare the intended usage of the key through a *key type* specification. The cryptographic subsystem then enforces this specification by denying requested services that are inappropriate for the declared key type. For example, a key that is used to encrypt data cannot be used to encrypt a key. Likewise, a key that is designated a key-encrypting key cannot be employed in a decryption operation, thereby preventing the use of a key-encrypting key to obtain a cleartext key.

Table 5-2 shows some of the key types supported by the CCA.

Table 5-2 Some CCA key types

Key type	Attributes
CIPHER	A 64-bit or 128-bit key used in the Encipher or Decipher callable service.
DATA	A 64-bit, 128-bit, or 192-bit key used in the Encipher, Decipher, MAC generate, or MAC verify callable service.
DATAC	A 128-bit key used in the Encipher or Decipher callable service, but not in the MAC generate or MAC verify callable service.
DATAM	128-bit key used in the MAC generate or MAC verify callable service.
DATAMV	128-bit key used in the MAC verify callable service.
DECIPHER	A 64-bit or 128-bit key used only to decrypt data. DECIPHER keys cannot be used in the Encipher callable service.
ENCIPHER	A 64-bit or 128-bit key used only to encrypt data. ENCIPHER keys cannot be used in the Decipher callable service.
EXPORTER	A 128-bit key-encrypting key used to convert a key from the operational form into exportable form.
IMPORTER	A 128-bit key-encrypting key used to convert a key from the importable form into operational form.
MAC	A 64-bit or 128-bit key used in the MAC generate or MAC verify callable service.
MACVER	A 64-bit or 128-bit key used in the MAC verify callable service but not in the MAC generate callable service.

Each type of key (except the master key) has a unique control vector associated with it. The bits in a control vector specify the possible uses of the key in great detail. For example, there are bits that specify the key type, the key subtype, whether the key can be exported, and

whether the key can be used in encryption, decryption, MAC generation, and MAC verification. This prevents the many attacks that are otherwise possible by using a key for an inappropriate function.

Whenever the master key is used to encrypt a key, the cryptographic hardware produces a variation of the master key according to the type of key that is being enciphered. These variations are called *master key variants*. The cryptographic hardware creates a master key variant by exclusive ORing a control vector with the master key. For example, when the master key is used to encipher a DATA key, the cryptographic hardware produces the master key DATA variant by XORing the master key with the control vector for DATA keys. After creating the master key DATA variant, the cryptographic hardware encrypts the DATA key by using the master key DATA variant as the key for the encryption algorithm. See Figure 5-15.

## 5.15 DES encryption

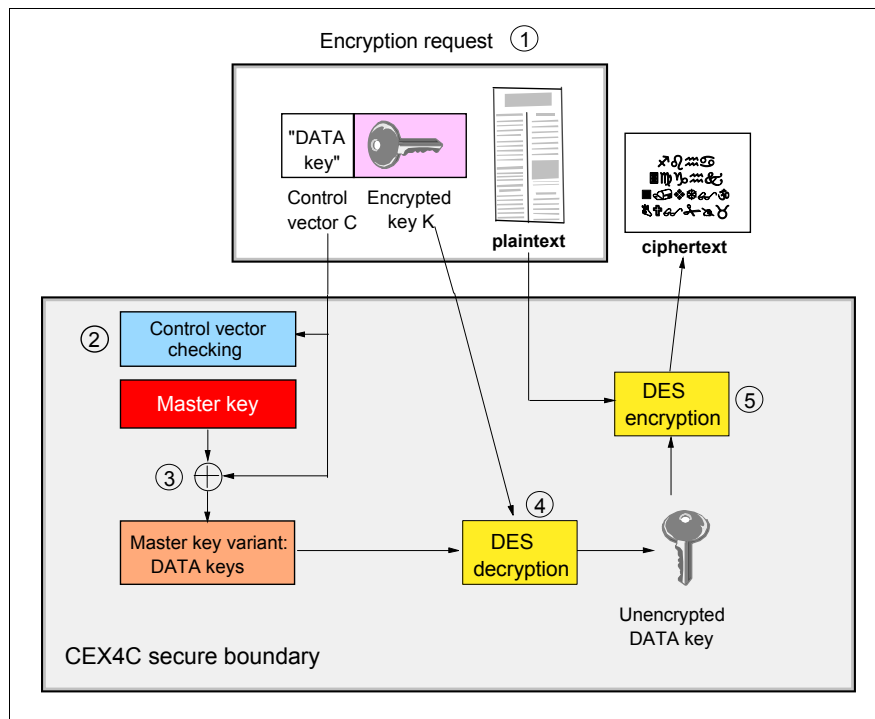


Figure 5-15 DES encryption

### DES encryption

In Figure 5-15, we formulate a request to encrypt some plaintext using a DATA key K that has already been encrypted under the SYM-MK master key of the CEX4C (1). K has an associated control vector C. C is examined to see if it has attributes that qualify it to be used in the called service in the requested way (2). If it does not, the service invocation fails. If C is valid, execution of the requested service continues. The CEX4C XORs the master key with the DATA Control Vector to produce a master key variant (3). Next, it uses the master key variant to decrypt our DATA key K (4). Finally, it performs the requested encryption using the decrypted DATA key (5).

Notice that each key K is encrypted in such a way that the value of the master key and the control vector C (associated with K) must be specified to recover the key.

If a caller alters the value of the control vector to permit use of the key in a command, the correct value of the key is not recovered by the key decryption process and any resulting output of the service is invalid, that is, any output is equivalent to that resulting from using a random unknown key value in that service.

DES key forms are shown in Figure 5-16.

## 5.16 DES key forms

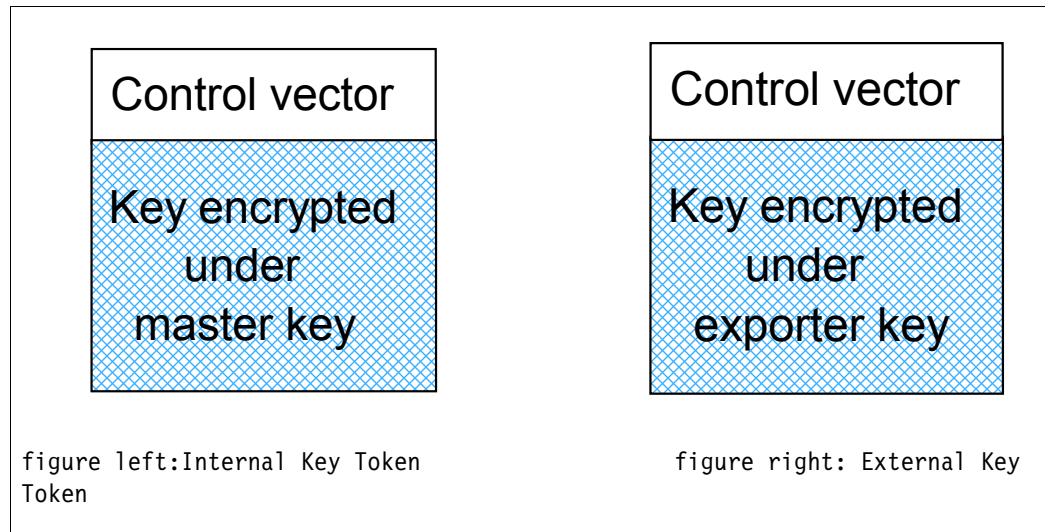


Figure 5-16 DES key forms

### DES key forms

The CCA specifies that a DES key must be in one of three *forms*:

► Operational

An *operational* key is a key that is encrypted under the master key at a particular system and can be used in a service at that system.

► Exportable

An *exportable* key is a key that is encrypted under an exporter key-encrypting key. In this form, a key can be sent outside the system to another system. A key in exportable form cannot be used in a cryptographic function.

► Importable

An *importable* key is a key that is encrypted under an importer key-encrypting key. A key is received from another system in this form. A key in importable form cannot be used in a cryptographic function.

The conversion from one key form to another key form is considered to be a one-way flow: *importable* → *operational* → *exportable*. An operational key form cannot be turned back into an importable key form, and an exportable key form cannot be turned back into an operational or importable key form.

Operational keys are accessed either directly by value in an *internal key token* or indirectly by a key label.

► Internal key token

As shown in Figure 5-16 on page 254, an internal key token contains an encrypted cryptographic key and its associated control vector. It is typically used for a key with a short life, as for example, a key that is used for a session and is disposed of when the session is over.

► Key label

A key label indirectly identifies an internal key token stored in key storage. (An example of key storage in the z/OS environment is the ICSF Cryptographic Key Data Set, a VSAM data set often called the CKDS). An operational key is a candidate for being kept in key storage if it is a key with a long life, if it is appropriate to control access to this key, or if many users need access to this key.

The *key\_identifier* parameter, which is found in most of the cryptographic API callable services, allows the programmer to pass keys to the service either directly by value or indirectly through a key label.

A key in importable or exportable form is kept in an *external key token*. The external key token contains the encrypted key and its associated control vector; see Figure 5-16 on page 254.

Figure 5-17 on page 256 shows a key export.

## 5.17 Key distribution: Key export

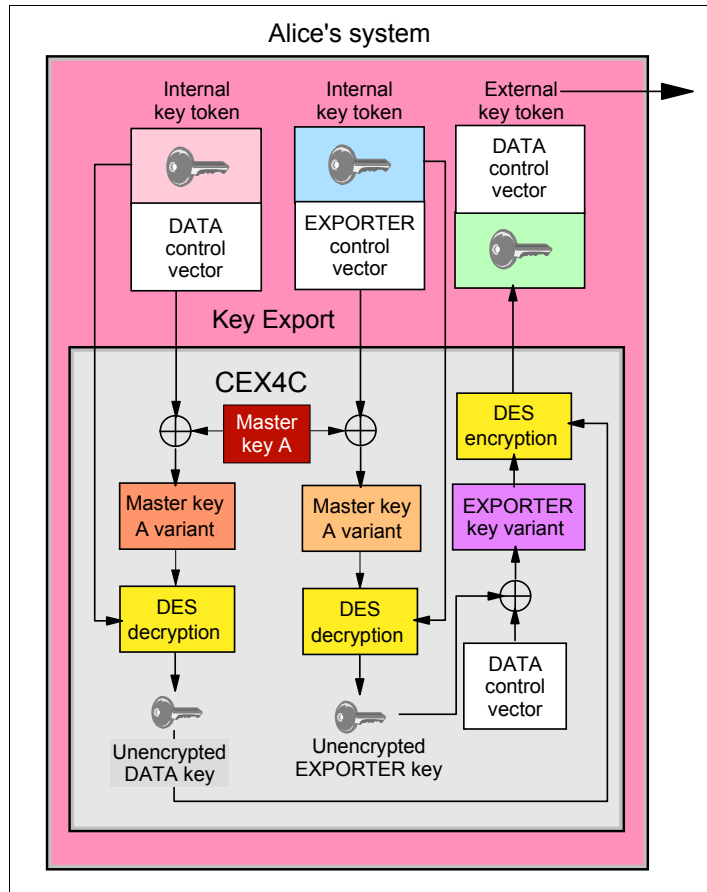


Figure 5-17 Key export

### Key distribution: Key export

The CCA uses the exportable and importable key forms to support electronic key distribution with minimal manual key installation. Suppose Alice wants to send a key *K* to Bob. An initial *exporter* key-encrypting key is installed on Alice's system by a courier, and an initial *importer* key-encrypting key is installed on Bob's system. The exporter key and the importer key have the *same value but different control vectors*.

After the manual installation of these initial key-encrypting keys, all subsequent key distribution can be done electronically. For example, Alice can execute the key export service to convert the information for *K* found in its internal key token to an exportable key in an external key token. The external key token contains *K* encrypted under the exporter key (instead of the master key) and *K<sub>s</sub>* that are associated control vector. The key is encrypted under the key-encrypting key that exists on Alice's sending system as an exporter key and on Bob's receiving system as an importer key. See where Alice sends a DATA key to Bob.

**Note:** Because the key-export service is performed in the CEX4C, the clear value of the key to be exported is not revealed. Also, note that if the content of the control vector is changed either accidentally or intentionally, the correct key value will not be recovered because the value of the encrypted key is cryptographically coupled to the control vector.

Figure 5-18 shows a key import.

## 5.18 Key distribution: Key import

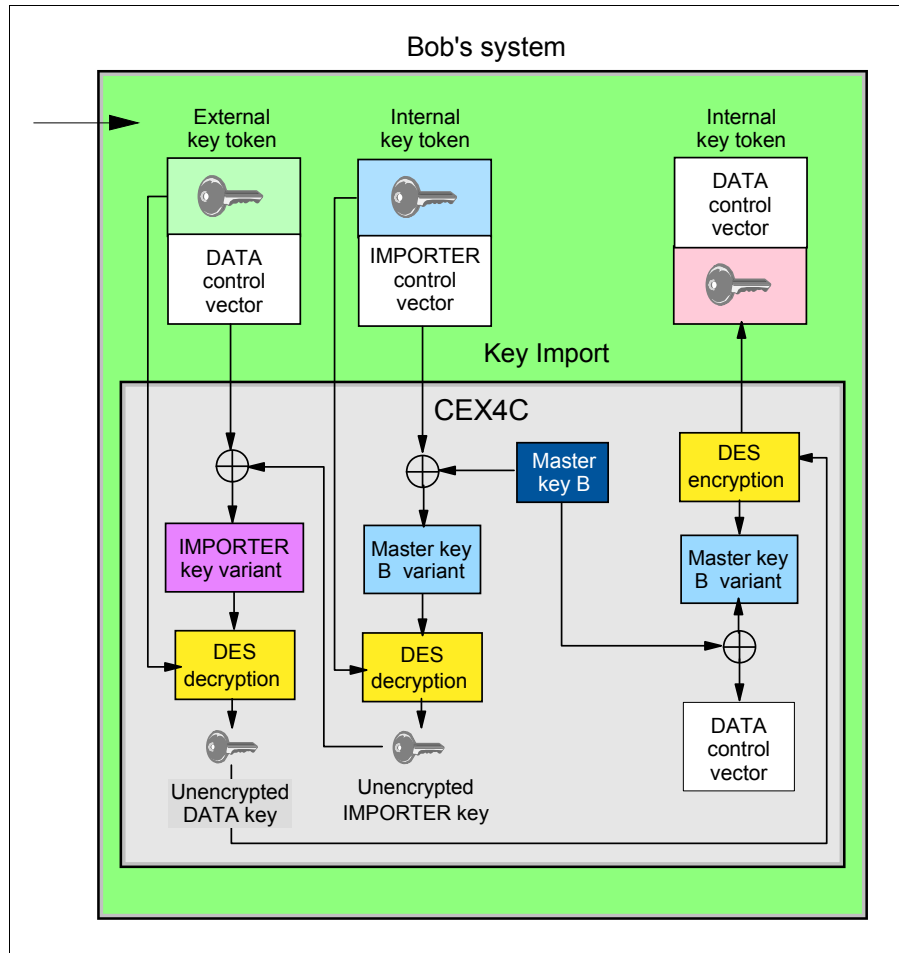


Figure 5-18 Key import

### Key distribution: Key import

Bob's system considers the key to be in importable form. An application on Bob's system can execute the key import service to perform the cryptographic transformations to convert the information in the external key token to an operational key in an internal key token. The intended usage of the key (that is, its type) is maintained through the control vector mechanism. When the key is re-enciphered from under the importer key to under the master key for Bob's system, it is in operational form and can be used again.

## 5.19 PKA key management

A public key algorithm (PKA) is an asymmetric cryptographic process in which a public key is used for encryption of secret (symmetric) keys and digital signature verification and a private key is used for decryption of secret keys and digital signature generation. RSA and DSA are two public key algorithms. The security of data protected by a PKA depends on the security of the private key. The CCA uses a master key to protect private keys. Private keys are active on a system only when they are encrypted under the master key, so the master key protects all private keys that are used on the system. A master key always remains in a secure area in the cryptographic hardware. In a z/OS environment, an ICSF administrator initializes and changes master keys using the ICSF panels or a Trusted Key Entry (TKE) workstation.

Almost all private keys that are encrypted under a master key are stored outside the protected area of the cryptographic hardware; they cannot be attacked because the master key used to encrypt them is itself secure inside the tamper-protected cryptographic hardware and will be zeroized if there is any attempted attack.

There is one exception to the rule that private keys are stored outside the cryptographic hardware. CCA supports *retained RSA keys*, in which the RSA key pair is generated inside the secure cryptographic hardware, and only the public key is ever allowed to leave the secure environment. The private key remains inside the secure hardware and is never allowed to leave in any form. This key is designed to meet the strict demands of some standards, which require assurance that the private key can exist only in a single cryptographic module. This rule greatly strengthens non-repudiation. If a private key can exist only in one cryptographic device, it provides assurance that any digital signature computed using that private key can have originated only at the system in which that device is installed. In the PCIXCC, retained RSA private keys are stored in the flash memory inside the secure module. Similar to all CCA data stored in that memory, they are securely encrypted under a TDES key that is destroyed if there is any attempt to tamper with the device.

Conceptually, the master key used to protect DES keys could have also been used to protect PKA private keys. However, the CCA designers chose to use a different master key as follows:

- ▶ When the cryptographic hardware is a PCICC or PCIXCC/CEX2C/CEX3C/CEX4C, the 192-bit master key is called the Asymmetric-keys Master Key (ASYM-MK).
- ▶ When the cryptographic hardware is a CCF, there are two PKA master keys:
  - The Key Management Master Key (KMMK) is a 192-bit key that is used to protect private keys that are used in both digital signature generation and decryption of secret (symmetric) keys.
  - The Signature Master Key (SMK) is a 192-bit key that is used to protect private keys that are used *only* in digital signature generation.



## Key forms

As was the case with DES keys, the CCA specifies that a PKA private key must be in one of three forms:

- ▶ Operational

An *operational* private key is a key that is encrypted under a PKA master key at a particular system and can be used in a service at that system.

- ▶ Exportable

An *exportable* private key is a key that is either in cleartext or is encrypted under a DES exporter key-encrypting key. In this form, a key can be sent outside the system to another system. A private key in exportable form cannot be used in a cryptographic function.

- ▶ Importable

An *importable* private key is a key that is either in cleartext or is encrypted under a DES importer key-encrypting key. A key is received from another system in this form. A private key in importable form cannot be used in a cryptographic function.

Operational keys are accessed either directly by value in an *internal key token* or indirectly by a key label:

- ▶ *Internal key token*

The format of an RSA private internal key token differs from the format of a DSS private internal key token; we only discuss the former. As shown in Figure 5-19, an RSA private internal key token contains several sections:

- *R* indicates that the section is required
- *O* indicates that the section is optional

In Figure 5-19 and succeeding figures:

- *d* represents the RSA private exponent
- *e* represents the public exponent
- *n* represents the modulus

Token identifier: X'1F'	Header (R)
	RSA private key section (R)
Public key modulus length in bits Public key exponent e	RSA public key section (R)
	RSA private key name (O)
Flag byte indicating whether: RSA or DSS key Private or public key Private key name section exists Private key is unenciphered Key is a retained key Count of number of sections Info about key if it is retained	Internal information section (R)

Figure 5-19 RSA private key: Internal key token

An access control system can use the private key name to verify that the calling application is entitled to use the key.

The RSA private key section can have three forms:

- 1024-bit modulus exponent form for the CCF.
- 2048-bit Chinese Remainder Theorem form.
- 1024-bit modulus exponent form for the PCICC, PCIXCC, CEX2C, CEX3C, or CEX4C. See Figure 5-20.

SHA-1 hash value of the next sub-section. This hash value is checked after an enciphered private key is deciphered for use.	Blinding information sub-section
Key use flag bits: Decryption of secret keys permitted Digital signature generation permitted Object protection key (OPK) encrypted under the ASYM-MK Private exponent d encrypted under the OPK Modulus n	
SHA-1 hash value of the blinding information sub-section	
Random number r Random number r <sub>1</sub> X'00' padding to get a multiple of 8 bytes	

Figure 5-20 1024-bit modulus exponent form for CEX2C

► Key label

A *key label* indirectly identifies an internal key token stored in key storage. (An example of key storage in the z/OS environment is the ICSF Public Key Data Set, a VSAM data set often called the PKDS).

The **key\_identifier** parameter found in most of the cryptographic API callable services allows the programmer to pass keys to the service either directly by value or indirectly through a key label.

A private key in importable or exportable form is kept in an *external key token*. The format of an RSA private external key token differs from the format of a DSS private external key token; we only discuss the former. As shown in Figure 5-21, an RSA private external key token contains several sections. Again, *R* indicates that the section is required and *O* indicates that the section is optional.

Token identifier: X'1E'	Header (R)
	RSA private key section (R)
Public key modulus length in bits Public key exponent e	RSA public key section (R)
	RSA private key name (O)

Figure 5-21 RSA private key: external key token

The RSA private key section can have two forms:

- 1024-bit modulus exponent form for the CCF and PCICC.
- 2048-bit Chinese Remainder Theorem form for the PCICC, PCIXCC, or CEX2C. See Figure 5-22.

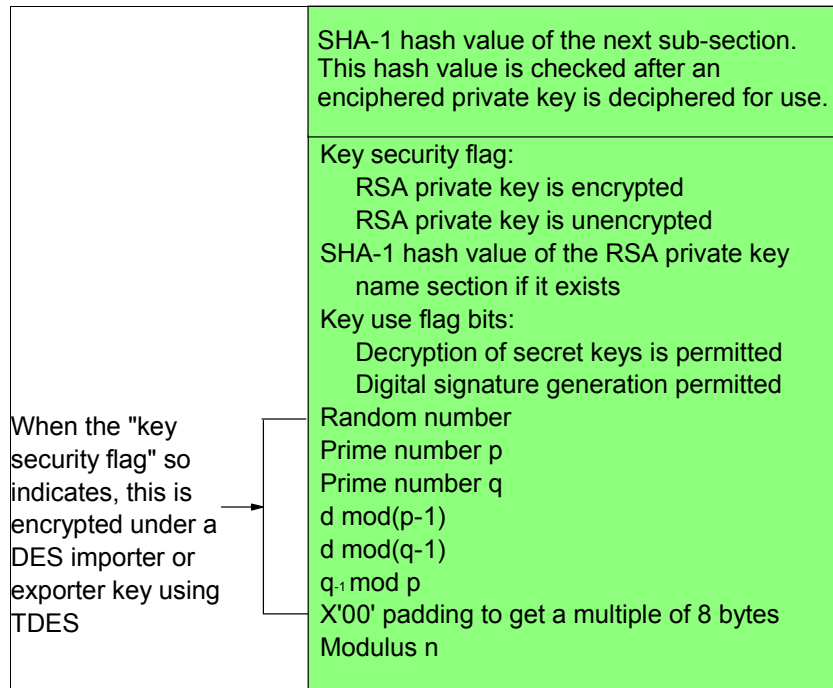


Figure 5-22 Chinese Remainder Theorem form

You can use the PKA Key Import callable service to do either of the following tasks:

- ▶ Get a private key deciphered from an importer key and enciphered by the ASYM-MK.
- ▶ Get a clear, unenciphered private key enciphered by the ASYM-MK.

So far we have only discussed tokens for RSA *private* keys. The CCA also defines a token for RSA public keys. Because public keys are meant to be shared, the format of an RSA public key token is rather simple:

- ▶ Header containing a token identifier of X'1E' (indicating an external token)
- ▶ RSA public key section containing the public exponent e and the modulus n in cleartext.

CCA callable services can use PKA public key tokens directly in the external form.

Figure 5-23 on page 262 provides a schematic view of the hardware cryptography implementation in the System z environment.

## 5.20 Integrated Cryptographic Service Facility

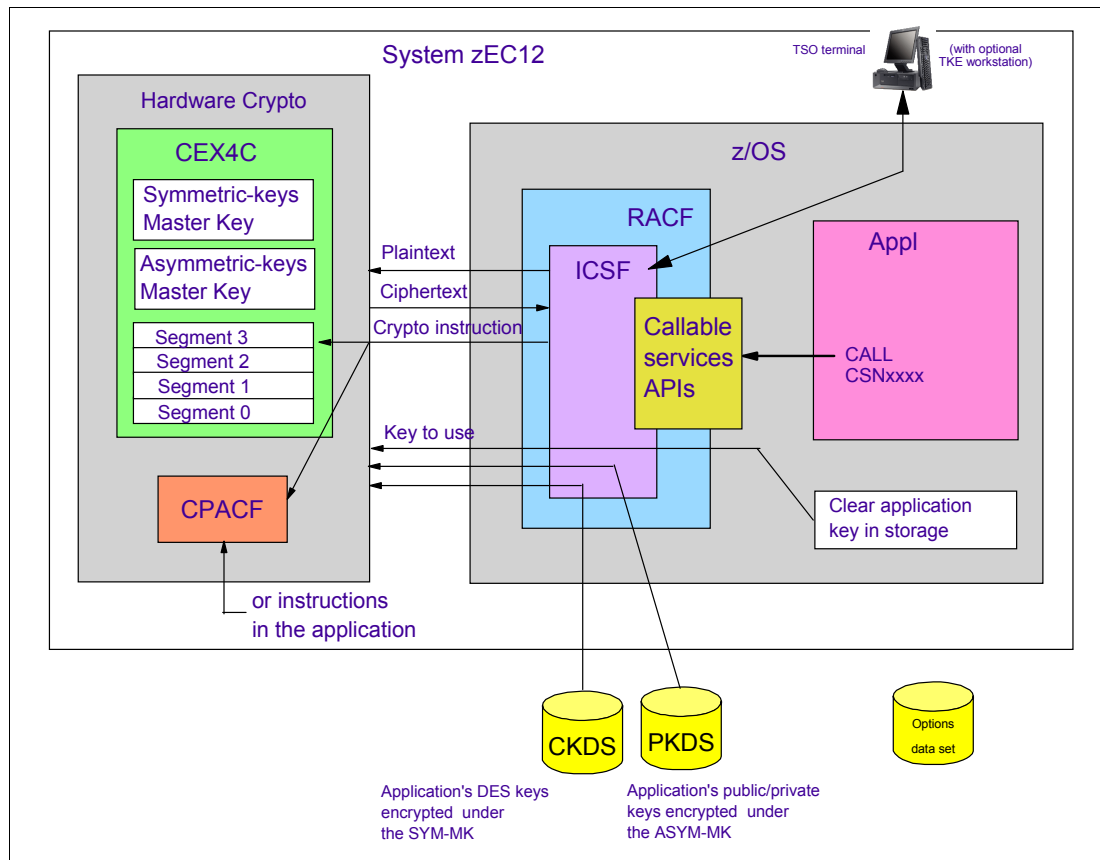


Figure 5-23 Integrated Cryptographic Service Facility

### Integrated Cryptographic Service Facility

In the z/OS environment, it is the Integrated Cryptographic Service Facility (ICSF) that provides access to cryptographic functions through callable services. The ICSF callable services comply with the IBM CCA cryptographic API and are available for programs written in assembly language or high-level languages. IBM CCA supports a hierarchical structure of keys, where keys can be encrypted by other keys (key-encrypting keys, KEKs), the master key being at the top of the hierarchy.

ICSF provides cryptographic coprocessors administration facilities for those coprocessors that require a master key to be set.

ICSF also provides key repositories in the form of two VSAM data sets, where keys can be kept in *key tokens* in clear value or encrypted under a KEK or under the coprocessors master keys. The VSAM data sets are the CKDS and the PKDS. The key tokens in the CKDS and the PKDS are given a user-defined or system-defined label that is used for their retrieval and maintenance.

**Note:** The hardware cryptography technology that we discuss here is available on the IBM System z9 and eServer zSeries 990 and 890 platforms. The zSeries 800 and 900 host other, although functionally compatible, types of cryptographic coprocessors.

In Figure 5-23 on page 262, an application program has issued a CCA cryptographic API call on a System zEC12. The call is routed to the ICSF started task. The ICSF started task invokes RACF to determine whether the user ID associated with the request is authorized to use the requested cryptographic service and any keys associated with the request. If the user ID has the proper authority, the ICSF started task decides whether it should perform the request using ICSF software or cryptographic hardware.

If ICSF decides to use cryptographic hardware, it gives control to its routines that contain the crypto instructions. (The cryptographic instructions that drive the CPACF are listed in 5.11, “CP Assist for Cryptographic Functions” on page 247.) ICSF routes the request to the CEX4C and if the request is, say, a request to encrypt data, the ICSF started task provides the CEX4C with the data to be encrypted and the key to be used by the encryption algorithm. Recall that the key is encrypted, in this case under a variant of the SYM-MK stored in the CEX4C. The request proceeds as shown previously in Figure 5-15 on page 253.

The interactions between the functional blocks shown in Figure 5-23 on page 262 are as follows:

- ▶ ICSF is a z/OS started task that offers cryptographic APIs to applications and drives the requests to the Crypto Express4S Coprocessor (CEX4C).
- ▶ The CEX4C is a “secure” coprocessor in that it contains a master key used to encrypt keys to be kept in storage or in the PKDS data set. The master key resides in the coprocessor hardware only and is used to decrypt internally to the coprocessor the secure keys that are provided so that they can be used to encrypt or decrypt data.
- ▶ ICSF needs other data sets to operate. The CKDS for the use of cryptographic hardware, and an options data set that contains the ICSF started task startup parameters. ICSF requires a PKDS as well. The PKDS does not need to contain any records, or even be initialized, but it does need to be allocated by ICSF.
- ▶ Installing and maintaining the secret master key is a task that security officers can perform from TSO/E terminals or from an optional TKE workstation, the latter for a very high security level of the interactions between the security officers and the CEX4C.

If there is more than one secure coprocessor to which ICSF has access, all coprocessors must have been set with the same master key value.

- ▶ The CPACF operates only with clear keys.

The keys can be stored in ICSF-managed VSAM data sets and pointed to by the application program by using the label under which they are stored. The CKDS is used to store the symmetric keys in their encrypted form, and the PKDS is used to store the asymmetric keys. If the level of ICSF that you are using is HCR7720 or higher, you can also store keys in the CKDS in clear (unencrypted) form.



# Related publications

We consider the publications that we list in this section particularly suitable for a more detailed discussion of the topics that we cover in this book.

## IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 267. Note that some of the documents referenced here might be available in softcopy only.

The other volumes in this series include:

- ▶ *ABCs of z/OS System Programming Volume 1*, SG24-6981  
Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
- ▶ *ABCs of z/OS System Programming Volume 2*, SG24-6982  
Implementing z/OS and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, Language Environment, and SMP/E
- ▶ *ABCs of z/OS System Programming Volume 3*, SG24-6983  
Introduction to DFSMS, data set basics, storage management hardware and software, VSAM, system-managed storage, catalogs, and DFSMSStvs
- ▶ *ABCs of System Programming Volume 4*, SG24-5654  
Communication Server, TCP/IP, and VTAM  
This volume is not yet published.
- ▶ *ABCs of z/OS System Programming Volume 5*, SG24-6985  
Base and Parallel Sysplex, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), and Geographically Dispersed Parallel Sysplex (GPDS)
- ▶ *ABCs of z/OS System Programming Volume 7*, SG24-6987  
Printing in a z/OS environment, Infoprint Server, and Infoprint Central
- ▶ *ABCs of z/OS System Programming Volume 8*, SG24-6988  
An introduction to z/OS problem diagnosis
- ▶ *ABCs of z/OS System Programming Volume 9*, SG24-6989  
z/OS UNIX System Services
- ▶ *ABCs of z/OS System Programming Volume 10*, SG24-6990  
Introduction to z/Architecture, System z processor design, System z connectivity, LPAR concepts, HCD, and HMC
- ▶ *ABCs of z/OS System Programming Volume 11*, SG24-6327  
Capacity planning, performance management, WLM, RMF, and SMF

Other publications of interest include:

- ▶ *System z Cryptographic Services and z/OS PKI Services*, SG24-7470
- ▶ *z9-109 Crypto and TKE V5 Update*, SG24-7123
- ▶ *Implementing PKI Services on z/OS*, SG24-6968
- ▶ *z/OS Version 1 Release 8 RACF Implementation*, SG24-7248

## Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS Integrated Security Services Network Authentication Service Administration*, SC23-6786
- ▶ *z/OS Integrated Security Services Network Authentication Service Programming*, SC23-6787
- ▶ *z/OS Cryptographic Services PKI Services Guide and Reference*, SA23-2286
- ▶ *z/OS Security Server RACF Auditor's Guide*, SA23-2290
- ▶ *z/OS Security Server RACF Callable Services*, SA23-2293
- ▶ *z/OS Security Server RACF Command Language Reference*, SA23-2292
- ▶ *z/OS Security Server RACF Data Areas*, GA32-0885
- ▶ *z/OS Security Server RACF Diagnosis Guide*, GA32-0886
- ▶ *z/OS Security Server RACF General User's Guide*, SA23-2298
- ▶ *z/OS Security Server RACF Macros and Interfaces*, SA23-2288
- ▶ *z/OS Security Server RACF Messages and Codes*, SA23-2291
- ▶ *z/OS Security Server RACF Security Administrator's Guide*, SA23-2289
- ▶ *z/OS Security Server RACF System Programmer's Guide*, SA23-2287
- ▶ *z/OS Security Server RACROUTE Macro Reference*, SA23-2294
- ▶ *z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference*, SA23-2297
- ▶ *z/OS OCSF Service Provider Module Developer's Guide and Reference*, SC14-7514
- ▶ *z/OS Open Cryptographic Services Facility (OCSF) Application Programming*, SC14-7513
- ▶ *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521
- ▶ *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SA22-7522
- ▶ *z/OS Cryptographic Services ICSF Messages*, SA22-7523
- ▶ *z/OS Cryptographic Services ICSF Overview*, SA22-7519
- ▶ *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520
- ▶ *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*, SC14-7511
- ▶ *z/OS Cryptographic Services System SSL Programming*, SC14-7495



## How to get IBM Redbooks publications

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

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





Redbooks

# ABCS of IBM z/OS System Programming Volume 6

(0.5" spine)  
0.475" x 0.873"  
250 <-> 459 pages







# ABCs of IBM z/OS System Programming Volume 6

## Security on z/OS

## RACF and SAF

## Cryptography

The ABCs of IBM z/OS System Programming is an 11-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects.

Following are the contents of the volumes:

- ▶ Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
- ▶ Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, IBM Language Environment, and SMP/E
- ▶ Volume 3: Introduction to DFSMS, data set basics, storage management hardware and software, VSAM, System-managed storage, catalogs, and DFSMSStvs
- ▶ Volume 4: Communication Server, TCP/IP, and IBM VTAM
- ▶ Volume 5: Base and IBM Parallel Sysplex, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), and IBM Geographically Dispersed Parallel Sysplex (IBM GDPS)
- ▶ Volume 6: Introduction to security, IBM RACF, digital certificates and PKI, Kerberos, cryptography and IBM z9 integrated cryptography, LDAP, and EIM
- ▶ Volume 7: Printing in a z/OS environment, Infoprint Server, and Infoprint Central
- ▶ Volume 8: An introduction to z/OS problem diagnosis
- ▶ Volume 9: z/OS UNIX System Services
- ▶ Volume 10: Introduction to IBM z/Architecture, IBM System z processor design, System z connectivity, LPAR concepts, HCD, and HMC
- ▶ Volume 11: Capacity planning, performance management, WLM, IBM RMF, and SMF

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

## BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

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

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

SG24-6986-01

ISBN 0738439800