z/OS

Security Server RACF System Programmer's Guide

SA22-7681-04

z/OS

Security Server RACF System Programmer's Guide

SA22-7681-04

Note

Before using this information and the product it supports, be sure to read the general information under Appendix E, "Notices," on page 367.

Fifth Edition, March 2004

This is a major revision of SA22–7681–03. This edition applies to Version 1 Release 5 of z/OS (5694-A01), Version 1 Release 5 of z/OS.e (5655–G52), and all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

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

© Copyright International Business Machines Corporation 1994, 2004. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures xi

Tables xiii

About this document xv
Intended audience xv
Where to find more information xv
Softcopy publications xv
RACF courses xvi
Using LookAt to look up message explanations xvi
Accessing z/OS licensed documents on the Internet xvii
IBM systems center publications xvii
Other sources of information xviii
IBM discussion areas xviii
Internet sources xviii
To request copies of IBM publications xix

Summary of changes xxi

Chapter 1. Security and the RACF database 1
Data processing security 1
How RACF meets security needs 1
Identifying and verifying users 2
Authorizing users to access resources 2
Controlling access to resources 2
Logging and reporting 2
Administering security 2
Basic RACF concepts 3
RACF and the operating system 3
The RACF database 4
Database templates 5
Multiple data set support 7
Backup RACF database 8
Shared RACF databases 8
Sharing RACF data without sharing a database 10
Creating a RACF database 10
Finding a location for the RACF database 11
Copying your database 11
Using DFSMSdss DEFRAG 13
DFSMS enhanced data integrity (EDI) 13
Monitoring the usable space in your RACF database 13

Chapter 2. Performance considerations 15
The RACF database 16
Selection of control unit and device 16
Shared RACF database 16
Multiple data sets 17
Database housekeeping 17
Creating backup RACF databases 17
Resident data blocks 19
RVARY SWITCH command 19
Auditing 20
Operands requiring the AUDITOR attribute 20

© Copyright IBM Corp. 1994, 2004

RACF commands 21
RACF utility programs 22
BLKUPD 22
IRRUT200 22
Failsoft processing 22
Erase-on-scratch 23
Installation-written exit routines 24
Using global access checking 24
The SETROPTS command 24
Using SETROPTS RACLIST and SETROPTS GENLIST 25
Using SETROPTS INITSTATS and SETROPTS STATISTICS 29
Identification, verification, and authorization of user IDs 31
User identification and verification 31
RACROUTE REQUEST=AUTH processing 32
RACROUTE REQUEST=FASTAUTH processing 33
Using generic profiles 33
Mapping UIDs to user IDs and GIDs to group names 34
z/OS UNIX System Services applications 34
Large profiles 34
Large groups 35
Universal groups 35

Chapter 3. RACF customization 37
Specifying RACF database options 37
The data set name table 37
The database range table 45
Specifying resource-class options 48
The class descriptor table (CDT) 48
The RACF router table 52
Password authentication options 54
The RACF DES algorithm 54
Using the masking algorithm 56
Using your own authentication algorithm 56
PassTicket authentication 56
How RACF processes the password or PassTicket 56
Changing the RACF report writer options (ICHRSMFI module) 57
Customizing the RACF remote sharing facility 59
Customizing the RACF/DB2 external security module 59

Chapter 4. Operating considerations 61
Enabling and disabling RACF 62
Enabling RACF 63
Disabling RACF 63
Dynamic parse 64
Syntax of the IRRDPI00 command 65
RACF authorization of the IRRDPI00 command 65
TSO/E authorization 65
Automating IRRDPI00 65
ACEEs and VLF considerations 67
Dependencies 67
Operation 68
Removing information from VLF 68
VLF considerations for mapping UIDs and GIDs 69
Dependencies 69
VLF considerations for caching user security packets (USPs) 69
Dependencies 69

The RACF subsystem 69
Activating the RACF subsystem. 70
Restarting the RACF subsystem 76
Restarting a function in the RACF subsystem 76
Stopping the RACF subsystem address space 78
Diagnosing problems in the RACF subsystem 79
RACF operator commands 79
Group tree in storage 79
Shared database considerations 80
Using the global resource serialization function 80
RACF ENQ resources 81
Using Application Identity Mapping. 83
Sysplex considerations 84
Sharing a database 85
Sysplex communication. 87
Enabling sysplex communication 89
System authorization facility (SAF). 93
The SAF router. 93
The SAF callable services router 94
Associating started procedures and jobs with user IDs 94
Methods for associating started procedures with RACF identities 96
The STARTED class. 97
The started procedures table (ICHRIN03) 98
Coding the started procedures module 98
Changing the ICHAUTAB module 102
Failsoft processing 102
General considerations 103
Impact on users 104
CICS considerations 105
CICS timeout value. 105
DFP considerations. 105
TSO considerations. 106
ISPF considerations 106
DB2 considerations. 106
DASD data sets 106
Using utilities on RACF-protected DASD data sets 107
Moving a RACF-indicated DASD data set between systems 110
Using access method services commands 113
DASD volumes 113
Scratching DASD data sets 114
Moving DASD volumes between systems. 114
UCBs above 16MB 114
Protecting tape data 114
Tape data protection and bypass label processing (BLP) 114
Considerations for unlabeled (NL) tapes 115
Using utilities on RACF-protected tape volumes and tape data sets 115
Moving tape volumes between systems 115
Moving multivolume tape data sets between systems 115
Multiple users per address space. 116
Restarting jobs 116
Panel driver interface 116
REXX RACVAR function 117
Installing the REXX RACVAR function 117
Using the REXX RACVAR function 117

Chapter 5. RACF remote sharing facility (RRSF) 119

Overview of the RACF remote sharing facility (RRSF) 120
Understanding the RRSF concepts 120
Overview of the RRSF function 121
The RRSF network 128
RRSF nodes 128
Connections between nodes 132
Workspace data sets 134
How a directed command travels through the network 137
Order considerations for directed commands and application updates . . . 140
Defining an RRSF environment 141
Preparing to configure an RRSF network 142
Configuring an RRSF network 152
Customizing and establishing security for RRSF 174
Examples of defining a remote sharing environment 176
Monitoring your remote sharing environment 183

Chapter 6. The RACF/DB2 external security module 187
Installing the RACF/DB2 external security module 188
Customizing the RACF/DB2 external security module (optional) 189
Choosing the class scope 190
Defining classes for the RACF/DB2 external security module (optional) 194
Assembling and link-editing the RACF/DB2 external security module . . . 195
RACF/DB2 external security module functions 196
The initialization function (XAPLFUNC = 1) 196
The authorization function (XAPLFUNC = 2) 197
The termination function (XAPLFUNC = 3) 198

Chapter 7. RACF database utilities 201
RACF internal reorganization of aliases utility program (IRRIRA00) 203
IRRIRA00 stage conversion 204
Diagnostic capability 206
Input for IRRIRA00 207
Output from IRRIRA00 207
RACF database initialization utility program (IRRMIN00) 208
Running IRRMIN00 when PARM=NEW is specified 209
Running IRRMIN00 when PARM=UPDATE is specified. 210
Running IRRMIN00 when PARM=ACTIVATE is specified 211
Diagnostic capability 211
Input for IRRMIN00 211
Output from IRRMIN00 212
RACF cross reference utility program (IRRUT100) 212
Diagnostic capability 214
The work data set 214
Using IRRUT100 215
RACF database verification utility program (IRRUT200) 218
Copying a data set in the RACF database 218
Diagnostic capability 219
Monitoring the capacity of the RACF database 220
Processing considerations for databases from other systems 220
Using IRRUT200 221
Utility control statements 223
Scanning the index blocks 224
BAM/allocation comparison 228
IRRUT200 return codes 232
RACF database split/merge/extend utility program (IRRUT400). 232
How IRRUT400 works 233

Using IRRUT400 to extend a database . . . 233
Copying a RACF database . . . 233
Repairing a RACF database . . . 235
Diagnostic capability . . . 236
Executing IRRUT400 . . . 236
IRRUT400 return codes . . . 241
IRRUT400 examples . . . 242
Utilities documented in other documents . . . 245
RACF database unload utility program (IRRDBU00) . . . 245
RACF remove ID utility (IRRRID00) . . . 245
RACF SMF data unload utility program (IRRADU00) . . . 245
BLKUPD command . . . 245
Data security monitor (DSMON) . . . 245
RACF report writer (RACFRW) . . . 245
RRSF VSAM file browser (IRRBRW00) . . . 246
RACFICE reporting tool . . . 246

Chapter 8. RACF installation exits . . . 247
Overview . . . 249
RACF exits report . . . 250
Extended addressing for exits . . . 250
Data set naming convention table . . . 251
Exits running in the RACF subsystem address space . . . 252
Possible uses of RACF exits . . . 253
Summary of installation-exit callers . . . 253
ACEE compression/expansion exits . . . 255
Range tables . . . 256
IRRACX01 . . . 258
IRRACX02 . . . 260
Command exits for specific commands . . . 261
ICHCNX00 processing . . . 262
ICHCCX00 processing . . . 265
Common command exit . . . 266
Controlling the exit routine through the dynamic exits facility . . . 266
Replacing the exit routine . . . 267
Exit routine environment . . . 267
Exit recovery . . . 267
Exit routine processing . . . 267
Programming considerations . . . 270
Entry specifications . . . 270
Return specifications . . . 270
Coded example of the exit routine . . . 271
New-password exit . . . 271
ICHPWX01 processing . . . 272
Possible use of the exit . . . 274
Password authentication exits . . . 275
ICHDEX01 . . . 276
ICHDEX11 . . . 278
RACROUTE REQUEST=AUTH exits . . . 279
Extended addressing . . . 279
Preprocessing exit (ICHRCX01) . . . 280
Postprocessing exit (ICHRCX02) . . . 281
Possible uses of the exits . . . 282
RACROUTE REQUEST=DEFINE exits . . . 283
Extended addressing . . . 284
Automatic direction of application updates . . . 284

Preprocessing exit (ICHRDX01) 284
Postprocessing exit (ICHRDX02) 285
RACROUTE REQUEST=FASTAUTH exits 286
Preprocessing exits (ICHRFX01 and ICHRFX03) 286
Postprocessing exits (ICHRFX02 and ICHRFX04) 290
Possible uses of the exits 295
RACROUTE REQUEST=LIST exits 296
Pre- and postprocessing exit (ICHRLX01) 297
Selection exit (ICHRLX02) 297
RACROUTE REQUEST=VERIFY(X) exits 298
Preprocessing exit (ICHRIX01) 299
Postprocessing exit (ICHRIX02) 300
RACF report-writer exit 301
ICHRSMFE processing 302
SAF router exits 302

Chapter 9. Recovery Procedures 303
Overview 304
Exit Routine Considerations. 304
TSO Considerations 305
The RVARY Command 305
Synchronization Considerations 307
Considerations for Issuing RVARY from the RACFRCVY Started Procedure 308
Failures on the RACF Database 309
Sample Recovery Procedures 309
Failures Using Sysplex Data Sharing 311
Read-Only Mode. 311
Non–Data Sharing Mode 312
Recovery Scenarios 312
Failures during RACF Command Processing 315
Commands That Do Not Modify User-Created RACF Profiles 316
Commands That Have Recovery Routines 316
Commands That Perform Single Operations. 317
Commands That Perform Multiple Operations 318
Recovering from Errors in Identity Mapping Profiles 319
Recovering from Errors with Application Identity Mapping 321
Commands That Are Propagated for RACF Sysplex Communication. . . . 322
Failures during RACF Manager Processing 324
Failures during System Operations on RACF-Protected Data Sets 325
Failures during SCRATCH or DELETE. 326
Failures during ALLOCATE or DEFINE 326
Failures during RENAME or ALTER. 326
Failures during EOV (Non-VSAM) 327
Failures in the RACF Subsystem Address Space 327
Recovering from RACF Parameter Library Problems 327
Recovering When a Task Stops 328
Recycling an RRSF Connection 329
Recovering from VSAM Errors on the RRSF Workspace Data Sets 329
The Last Resort—Shutting Down the RACF Subsystem Address Space 331

Chapter 10. Storage Estimates 333
RACF Database Storage Requirements 333
Factors Affecting the Size of the RACF Database. 333
Formula for the RACF Database Size 333
RACF Virtual Storage Requirements 336
Coupling Facility Cache Structure Storage Requirements 338

Appendix A. Supplied CDT classes 339
Supplied resource classes for z/OS and OS/390 systems 339
Supplied resource classes for z/VM and VM systems 346

Appendix B. RRSF Initialization Worksheet and Scenario 349
RRSF Node Configuration Worksheet 350
RRSF Initialization Scenario 351
Background Information 351
Completed RRSF Node Configuration Worksheet for Node MVS01 353
Completed RRSF Node Configuration Worksheet for Node MVS02 354
Summary 355
Detailed Instructions 355
Now It's Your Turn to Fill Out the Worksheet 360

Appendix C. Non-recommended Options 361
Selecting Options with ICHSECOP 361
Bypassing RACF-Initialization Processing. 361
Selecting the Number of Resident Data Blocks. 362
Disallowing Duplicate Names for Data Set Profiles 362
Changing the ICHAUTAB Module 363
Using the RACF Authorized-Caller Table 363

Appendix D. Accessibility 365
Using assistive technologies 365
Keyboard navigation of the user interface. 365
z/OS Information 365

Appendix E. Notices 367
Programming Interface Information 368
Trademarks. 369

Index 371

Figures

1. RACF and Its Relationship to the Operating System 4
2. ICHRDSNT example 1—for three data sets 43
3. ICHRDSNT example 2—data sharing option and split database 44
4. ICHRDSNT example 3—sysplex communication and split database 45
5. ICHRRNG example for three data sets 48
6. Setting Up the RACF Subsystem Address Space 71
7. RESTART Command Syntax 76
8. RACF STOP Command Syntax 78
9. Coding ICHRIN03 So the Assembler Calculates the Count Field 99
10. An RRSF network 128
11. An RRSF network containing a single-system node and a multisystem node 130
12. An RRSF network with two multisystem nodes and one single-system node 131
13. A directed command traveling through an RRSF network 138
14. SET command syntax 153
15. Sample output from the SET LIST command 154
16. TARGET command syntax 156
17. Summary information displayed by a TARGET LIST command for a single-system node 162
18. Detailed information displayed by a TARGET LIST command for a single-system node 163
19. Summary information from the TARGET LIST command for a multisystem node 163
20. Detailed information from the TARGET LIST command for a system on a multisystem node 164
21. Example of a RACF parameter library for a node running in local mode 171
22. Example of a RACF parameter library for a node running in remote mode 172
23. Two RRSF nodes in local mode 177
24. An RRSF network with two nodes 178
25. An RRSF network containing a multisystem node and a single-system node 179
26. An RRSF network containing two multisystem nodes 182
27. Default values for settable options 190
28. Single subsystem scope classes 191
29. Multi-subsystem scope classes 193
30. Sample output from IRRUT100 217
31. Sample output of formatted index produced by IRRUT200 226
32. Sample output of formatted alias index produced by IRRUT200 227
33. Sample output of encoded BAM map produced by IRRUT200 231
34. RACF Storage Use 336

© Copyright IBM Corp. 1994, 2004

Tables

1. Alias index entry values 47
2. Format of ICHRSMFI 58
3. RACF ENQ Resources 81
4. Some IBM products that use RACROUTE or ICHEINTY to update the RACF database 127
5. Connection States between Nodes 133
6. Defining an RRSF environment—summary of tasks 141
7. Displaying list information with the TARGET command 162
8. RRSFDATA resource names 175
9. DB2 object abbreviations 195
10. RACF utilities described in this chapter 202
11. IRRIRA00 stage summary 204
12. RACF installation-exits cross-reference table—Part 1 of 2 253
13. RACF installation-exits cross-reference table—Part 2 of 2 254
14. ICHCNX00-exit parameter processing. 263
15. Fields available during ICHPWX01 processing 273
16. Formula for the RACF Database Size. 333
17. RACF Estimated Storage Usage 337
18. Resource Classes for z/OS and OS/390 Systems 339
19. Resource Classes for z/VM and VM Systems 346

© Copyright IBM Corp. 1994, 2004

About this document

This document supports z/OS® (5694–A01) and z/OS.e (5655–G52).

This document contains information about the Resource Access Control Facility (RACF®), which is part of the Security Server for z/OS.

Intended audience

This publication is intended for MVS™ system programmers or MVS installation personnel responsible for:

- Maintaining RACF databases
- Writing, testing, and installing RACF exits
- Modifying the RACF program product to satisfy an installation's particular needs

You should be familiar with the information in:

- *z/OS Security Server RACF Migration*
- *z/OS Security Server RACF Security Administrator's Guide*
- The program directory shipped with z/OS

z/OS Security Server RACF Auditor's Guide, which describes the RACF report writer, might also be useful.

You should also be familiar with MVS and the z/OS library. In addition, if you use the RACF sysplex communication option, you should be familiar with the Parallel Sysplex® library. If you use the RACF remote sharing facility (RRSF), you should be familiar with APPC and VTAM®.

Where to find more information

Where necessary, this document references information in other publications. For complete titles and order numbers for all elements of z/OS, see *z/OS Information Roadmap*.

Softcopy publications

The RACF library is available on the following CD-ROMs. The CD-ROM online library collections include Softcopy Reader™, which is a program that enables you to view the softcopy documents.

SK3T-4269 *z/OS Version 1 Release 5 Collection*

This collection contains the set of unlicensed documents for the current release of z/OS in both BookManager® and Portable Document Format (PDF) files. You can view or print the PDF files with an Adobe Reader.

SK3T-4272 *z/OS Security Server RACF Collection*

This softcopy collection kit contains the Security Server library for z/OS in both BookManager and Portable Document Format (PDF) files. You can view or print the PDF files with an Adobe Reader.

SK2T-2180 *Online Library OS/390 Security Server RACF Information Package*

This softcopy collection contains the Security Server library for OS/390. It also contains the RACF/MVS Version 2 product libraries, the RACF/VM 1.10 product library, product documents from the

© Copyright IBM Corp. 1994, 2004

OS/390® and VM collections, International Technical Support Organization (ITSO) documents (known as Redbooks™), and Washington System Center (WSC) documents (known as orange books) that contain information related to RACF. The collection does not contain any licensed publications. By using this CD-ROM, you have access to RACF-related information from IBM® products such as OS/390, VM/ESA®, CICS®, and NetView®.

SK3T-7876 *IBM eServer zSeries™ Redbooks Collection*

This softcopy collection contains a set of documents called Redbooks that pertain to zSeries subject areas ranging from e-business application development and enablement to hardware, networking, Linux, solutions, security, Parallel Sysplex and many others.

SK2T-2177 *IBM Redbooks S/390® Collection*

This softcopy collection contains a set of documents called Redbooks that pertain to S/390 subject areas ranging from application development and enablement to hardware, networking, security, Parallel Sysplex and many others.

RACF courses

The following RACF classroom courses are available:

ES840 *Implementing RACF Security for CICS/ESA® and CICS/TS*

H3917 *Basics of OS/390 Security Server RACF Administration*

H3927 *Effective RACF Administration*

ES88A *Exploiting the Features of OS/390 Security Server RACF*

IBM provides a variety of educational offerings for RACF. For more information about classroom courses and other offerings, do any of the following:

- See your IBM representative
- Call 1-800-IBM-TEACH (1-800-426-8322)

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, and VSE:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e® systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Windows DOS command line.

- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

IBM systems center publications

IBM systems centers produce documents known as red and orange books that can help you set up and use RACF. These documents have not been subjected to any formal review nor have they been checked for technical accuracy, but they represent current product understanding (at the time of their publication) and provide valuable information on a wide range of RACF topics. They are not shipped with RACF; you must order them separately. A selected list of these documents follows. Other documents are available, but they are not included in this list, either because the information they present has been incorporated into IBM product manuals or because their technical content is outdated.

G320-9279	*Systems Security Publications Bibliography*
GG22-9396	*Tutorial: Options for Tuning RACF*

1. z/OS.e customers received a Memo to Licensees, (GI10-0684) that includes this key code.

GG24-3378	*DFSMS and RACF Usage Considerations*
GG24-3451	*Introduction to System and Network Security: Considerations, Options, and Techniques*
GG24-3524	*Network Security Involving the NetView Family of Products*
GG24-3970	*Elements of Security: RACF Overview - Student Notes*
GG24-3971	*Elements of Security: RACF Installation - Student Notes*
GG24-3972	*Elements of Security: RACF Advanced Topics - Student Notes*
GG24-3984	*RACF Macros and Exit Coding*
GG24-4282	*Secured Single Signon in a Client/Server Environment*
GG24-4453	*Enhanced Auditing Using the RACF SMF Data Unload Utility*
GG26-2005	*RACF Support for Open Systems Technical Presentation Guide*
GC28-1210	*System/390® MVS Sysplex Hardware and Software Migration*
SG24-4704	*OS/390 Security Services and RACF-DCE Interoperation*
SG24-4820	*OS/390 Security Server Audit Tool and Report Application*
SG24-5158	*Ready for e-business: OS/390 Security Server Enhancements*
SG24-5339	*The OS/390 Security Server Meets Tivoli®: Managing RACF with Tivoli Security Products*

Other sources of information

IBM provides customer-accessible discussion areas where RACF may be discussed by customer and IBM participants. Other information is also available through the Internet.

IBM discussion areas

IBM provides *ibm.servers.mvs.racf* newsgroup for discussion of RACF-related topics. You can find this newsgroup on news (NNTP) server *news.software.ibm.com* using your favorite news reader client.

Internet sources

The following resources are available through the Internet to provide additional information about the RACF library and other security-related topics:

- **Online library**

  To view and print online versions of the z/OS publications, use this address:

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

- **Redbooks**

  The documents known as Redbooks that are produced by the International Technical Support Organization (ITSO) are available at the following address:

  `http://www.ibm.com/redbooks/`

- **Enterprise systems security**

  For more information about security on the S/390 platform, OS/390, and z/OS, including the elements that comprise the Security Server, use this address:

  `http://www.ibm.com/servers/eserver/zseries/zos/security/`

- **RACF home page**

  You can visit the RACF home page on the World Wide Web using this address:

  `http://www.ibm.com/servers/eserver/zseries/zos/racf/`

- **RACF-L discussion list**

  Customers and IBM participants may also discuss RACF on the RACF-L discussion list. RACF-L is not operated or sponsored by IBM; it is run by the University of Georgia.

  To subscribe to the RACF-L discussion and receive postings, send a note to:

listserv@listserv.uga.edu

Include the following line in the body of the note, substituting your first name and last name as indicated:

```
subscribe racf-l first_name last_name
```

To post a question or response to RACF-L, send a note, including an appropriate `Subject:` line, to:

```
racf-l@listserv.uga.edu
```

- **Sample code**

  You can get sample code, internally-developed tools, and exits to help you use RACF. This code works in our environment, at the time we make it available, but is not officially supported. Each tool or sample has a README file that describes the tool or sample and any restrictions on its use.

  To access this code from a Web browser, go to the RACF home page and select the "Downloads" topic from the navigation bar, or go to ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/.

  The code is also available from `ftp.software.ibm.com` through anonymous FTP. To get access:

  1. Log in as user **anonymous**.
  2. Change the directory, as follows, to find the subdirectories that contain the sample code or tool you want to download:

     ```
     cd eserver/zseries/zos/racf/
     ```

  An announcement will be posted on the RACF-L discussion list and on newsgroup *ibm.servers.mvs.racf* whenever something is added.

  Note: Some Web browsers and some FTP clients (especially those using a graphical interface) might have problems using `ftp.software.ibm.com` because of inconsistencies in the way they implement the FTP protocols. If you have problems, you can try the following:

  – Try to get access by using a Web browser and the links from the RACF home page.
  – Use a different FTP client. If necessary, use a client that is based on command line interfaces instead of graphical interfaces.
  – If your FTP client has configuration parameters for the type of remote system, configure it as UNIX instead of MVS.

Restrictions

Because the sample code and tools are not officially supported,

– There are no guaranteed enhancements.
– No APARs can be accepted.

To request copies of IBM publications

Direct your request for copies of any IBM publication to your IBM representative or to the IBM branch office serving your locality.

There is also a toll-free customer support number (1-800-879-2755) available Monday through Friday from 8:30 a.m. through 5:00 p.m. Eastern Time. You can use this number to:

- Order or inquire about IBM publications
- Resolve any software manufacturing or delivery concerns
- Activate the program reorder form to provide faster and more convenient ordering of software updates

Summary of changes

Summary of changes
for SA22-7681-04
z/OS Version 1 Release 5

This document contains information previously presented in *z/OS Security Server RACF System Programmer's Guide*, SA22-7681-03, which supports z/OS Version 1 Release 4.

New information

- Chapter 1, "Security and the RACF database," on page 1 includes a new section discussing database templates.
- Chapter 1, "Security and the RACF database," on page 1 includes a new section discussing DFSMS enhanced data integrity (EDI), and documents that you do not need to exempt the RACF database from enhanced data integrity processing.

Changed information

- In Chapter 1, "Security and the RACF database," on page 1, the description of creating a RACF database has been updated to reflect that the SYSTEMP DD statement is no longer required.
- In Chapter 4, "Operating considerations," on page 61, the section on dynamic parse has been updated to mention using IRRMIN00 with the new PARM=ACTIVATE option instead of IPLing to activate a new set of database templates.
- In Chapter 4, "Operating considerations," on page 61, the list of classes that can cause information to be removed from VLF has been updated to include the SERVAUTH class, and the FACILITY class when the SETROPTS MLS option is active.
- In Chapter 4, "Operating considerations," on page 61, a new ENQ name has been added to the table of RACF ENQ resources.
- In Chapter 5, "RACF remote sharing facility (RRSF)," on page 119, an instruction to IPL after after running IRRMIN00 has been removed.
- In Chapter 5, "RACF remote sharing facility (RRSF)," on page 119, an example of output from the SET LIST command has been updated to show new information that is displayed for the template version.
- Chapter 6, "The RACF/DB2 external security module," on page 187 has been updated to note that the chapter applies only to DB2® Version 7 and below.
- In Chapter 6, "The RACF/DB2 external security module," on page 187, the View object has been added to the table of DB2 object abbreviations.
- In Chapter 7, "RACF database utilities," on page 201, the section on IRRMIN00 has been updated. Information has been added about the new PARM=ACTIVATE option on IRRMIN00, which allows you to activate a new set of database templates without an IPL. This section also documents that the database templates are now shipped in the CSECT IRRTEMP2, that the PARM=NEW option on IRRMIN00 now prevents you from accidentally reinitializing an active RACF data set, that the PARM=UPDATE option automatically applies the latest database templates, that the SYSTEMP DD statement is no longer required for IRRMIN00, and new return codes from IRRMIN00.
- In Chapter 7, "RACF database utilities," on page 201, a note has been added to the description of the job control statements for the IRRUT200 utility documenting

© Copyright IBM Corp. 1994, 2004

that you should not specify the RLSE subparameter with the SPACE subparameter on the SYSUT1 DD statement.

- In Chapter 8, “RACF installation exits,” on page 247, in the section on the common command exit, text describing how the image of the command passed to the exit might differ from the original command has been expanded. The section titled ″Considerations for mixed-case names″ has been deleted, because the inconsistencies documented in this section no longer occur.
- Chapter 10, “Storage Estimates,” on page 333 has been updated to reflect changes to storage requirements for RACF for z/OS Version 1 Release 5.

Deleted information

- In “About this document” on page xv, the list of the components of the Security Server for z/OS has been removed. RACF is now the only component of the Security Server.

This document includes terminology, maintenance, and editorial changes. Technical changes or additions to the text are indicated by a vertical line to the left of the change.

Summary of changes
for SA22-7681-03
z/OS Version 1 Release 4

This document contains information previously presented in *z/OS Security Server RACF System Programmer's Guide*, SA22-7681-02, which supports z/OS Version 1 Release 3.

New information

- Information is added to indicate that this document supports z/OS.e
- The LDAPBIND class been added
- The AUTOUID and AUTOGID (for the ADDGROUP, ADDUSER, ALTGROUP, and ALTUSER commands) keywords have been added
- “Failsoft mode” on page 89 has been added to discuss the emergency mode called ″failsoft mode″
- Sysplex considerations have been added for specifying RACF database options
- Sysplex recovery scenarios have been added for considering recovery scenarios

Changed information

- Table 3 on page 81 has new entries
- IEBGENER considerations have been added for copying a data set in the RACF database

Starting with z/OS Version 1 Release 2, you might notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document includes terminology, maintenance, and editorial changes.

Summary of changes
for SA22-7681-02
z/OS Version 1 Release 3

This document contains information previously presented in *z/OS Security Server RACF System Programmer's Guide*, SA22-7681-01, which supports z/OS Version 1 Release 2.

New information

- The list of RACF ENQ resources in Chapter 4 has been updated to include cache information that supports Policy Director Authorization Services.
- An appendix with z/OS product accessibility information has been added.

Changed information

The factors that affect the size of the RACF database in Chapter 10 has been updated to consider the CACHECLS.

Deleted information

- Instructions for updating the `SCHEDxx` member of SYS1.PARMLIB was deleted because it is no longer valid.
- The glossary has been moved to *z/OS Security Server RACF Security Administrator's Guide*.

This document includes terminology, maintenance, and editorial changes. Technical changes or additions to the text are indicated by a vertical line to the left of the change.

Starting with z/OS Version 1 Release 2, you might notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes
for SA22-7681-01
z/OS Version 1 Release 2

This document contains information previously presented in *z/OS Security Server RACF System Programmer's Guide*, SA22-7681-00, which supports z/OS Version 1 Release 1.

The following summarizes the changes to that information.

New information

- Chapter 2 includes performance information for universal groups.
- Chapter 6 includes support for the Java™ archive object introduced in DB2 Version 7. This includes a description of the &ERROROPT set symbol and reason code 16 for the RACF/DB2 initialization function, XAPLFUNC.
- In Chapter 8, the password authentication exits, ICHDEX01 and ICHDEX11, have been updated to include return code 16.
- In Chapter 8, the common command exit, IRREVX01, has been updated with information about using mixed case names.

Changed information

- The description of shared database considerations in Chapter 4 has been updated to include VLF cache information.

- The description of RACF dynamic parse information in Chapter 5 has been modified.
- In Chapter 7, the IRRMIN00 utility has been updated with general information. The IRRUT100 utility has been updated to include information about universal groups.
- In Chapter 8, information has been updated about processing the new-password exit, ICHPWX01.
- In Chapter 10, RACF global access table estimates have moved to the ESCA storage subpool.

Deleted information

- IEBTCRIN has been removed from the list of RACF data sets in Chapter 4. It is no longer valid.

This document includes terminology, maintenance, and editorial changes.

Chapter 1. Security and the RACF database

Data processing security 1
How RACF meets security needs 1
Identifying and verifying users 2
Authorizing users to access resources 2
Controlling access to resources 2
Logging and reporting 2
Administering security 2
Basic RACF concepts 3
RACF and the operating system 3
The RACF database 4
Database templates 5
Keeping all copies of your database templates at the same level 5
Multiple data set support 7
Backup RACF database 8
Taking additional backup measures 8
Shared RACF databases 8
General considerations 9
Considerations for the SCICSTST and UCICSTST classes 9
RACGLIST considerations 10
RACF sysplex communication 10
Sharing RACF data without sharing a database 10
Creating a RACF database 10
Finding a location for the RACF database 11
Copying your database 11
Procedure for the primary database 12
Procedure for the backup database 12
Using DFSMSdss DEFRAG 13
DFSMS enhanced data integrity (EDI) 13
Monitoring the usable space in your RACF database 13

Data processing security

As the number of users and the ease of use of data systems increase, the need for data security takes on new importance. An installation can no longer ignore security simply because few people know how to access the data. Installations must actively pursue and demonstrate security and use a security mechanism to control any form of access to critical data.

How RACF meets security needs

RACF helps meet your needs for security by providing the ability to:

- 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.

© Copyright IBM Corp. 1994, 2004

A specific RACF user, called the security administrator, has the responsibility to define users and resources to RACF. The security administrator sets down the guidelines that RACF uses to decide the user-resource interaction within the installation.

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 writes, installs, and tests RACF installation exit routines.

RACF retains information about the users, resources, and access authorities in *profiles* on the *RACF database* and refers to the profiles when deciding which users should be permitted access to protected system resources.

Identifying and verifying users

RACF uses a *user ID* to identify the person who is trying to gain access to the system and the *password* to then verify the authenticity of that identity. RACF uses the concept of only one person knowing a particular user ID-password combination to verify user identities and to ensure personal accountability.

Authorizing users to access resources

Having identified and verified the user, RACF then controls interaction between the user and the system resources. RACF must authorize not only the users who may access resources, but also the way the user may access them, which depends on the user's purpose—reading, for example, or updating. RACF also can authorize *when* a user can access resources, by either time or day.

Controlling access to resources

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.

The installation establishes the controls; RACF enforces them.

Logging and reporting

Having identified and verified the user and limited access to resources, RACF records the events where attempted user-resource interaction has occurred. An installation can use logging and reporting to alert management not only to anticipated user activities and system events but also to variances from the expected use of the system.

Administering security

Because the security requirements at every data-processing installation differ, RACF allows an installation to meet its own, unique security objectives.

In many cases, it is easier to ignore security procedures than to use them. Even conscientious users can forget to protect a critical piece of data. The solution to the problem of implementing effective security measures is to provide a security system that is transparent to the user.

With RACF, end users do not need to be aware that a program is protecting their data. By making use of RACF's administrative capabilities, an installation can make the use of RACF transparent to most of its end users.

Basic RACF concepts

RACF can help meet an installation's security needs because it allows the installation to define *users* who can access protected *resources*, and, concurrently, to determine *how* users can access the protected resources.

With RACF, each defined user belongs to at least one group, known as the default group. A group is a collection of RACF users who share common access requirements to protected resources or who have similar attributes within the system.

RACF records information about the groups in the *group profile*, which resides in the RACF database.

RACF allows users to be members of more than one group. A RACF user who is associated with a group is, in RACF terminology, *connected* to that group.

A group owner—usually the user who defined the group to RACF—can define and control the other users connected to the group. The group owner can also delegate various group administrative responsibilities and authorities to various users connected to the group. RACF uses connect information in the user profile.

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

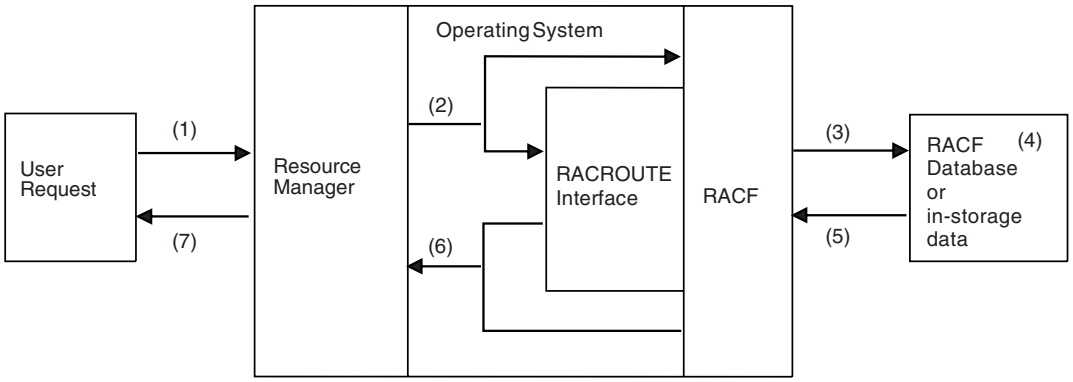
RACF and the operating system

RACF acts as a layer in the operating system.

For example:

1. A user is identified and verified to the RACF-protected system.
2. A user wants to modify an existing RACF-protected resource.
3. The user issues a command to the system to access the resource.
4. The system resource manager (such as data management) processes the request.
5. The resource manager "asks" RACF whether the user can access the resource.
6. 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.
7. RACF returns the results of its check to the resource manager.
8. The resource manager, based on what RACF indicates, either grants or denies the request.

Figure 1 on page 4 shows how RACF interacts with the operating system to allow access to a protected resource. The operating system-RACF interaction to identify and verify users is similar.



(1) User requests access to a resource using a resource manager (such as TSO/E, CICS, or IMS).

(2) The resource manager issues a RACF request to see if the user can access the resource. In most cases, this is a RACROUTE macro. In other cases, this is an independent RACF macro.

(3) RACF refers to the RACF database (or profiles copied into storage from the RACF database) and ...

(4) ...checks the appropriate resource profile.

(5) Based on the information in the profile...

(6) ...RACF passes the status of the request (the user can or cannot access the resource as intended) to the resource manager.

(7) The resource manager grants (or denies) the user request.

Figure 1. RACF and Its Relationship to the Operating System

During authorization checking, RACF ensures that a user has the authorization to access the requested protected resource. RACF checks the resource profile to ensure, for example, that the resource can be accessed in the way requested and that the user has the proper authorization to access the resource.

The RACF mechanism is analogous to the tumblers of a lock, all of which must align before the lock can open. In RACF, the necessary user-resource requirements must match before RACF grants the request to access a protected resource.

The RACF database

The RACF database holds all RACF access-control information. RACF processing uses the information from the database:

- Each time a RACF-defined user enters a system
- Each time a user wants to access a RACF-protected resource

You maintain your RACF database through commands, macros, and utilities.

The format of the database is described in *z/OS Security Server RACF Diagnosis Guide*.

The database templates are documented in *z/OS Security Server RACF Macros and Interfaces* and *z/OS Security Server RACROUTE Macro Reference*.

Information on protecting the RACF database is in *z/OS Security Server RACF Security Administrator's Guide*.

Information on estimating the size of the RACF database is in "RACF Database Storage Requirements" on page 333.

Database templates

The RACF database contains records whose format is controlled by a set of database templates. The templates map out how profiles are written on the RACF database. RACF ships the templates in a CSECT named IRRTEMP2. The format of the database templates is documented in *z/OS Security Server RACF Macros and Interfaces* and *z/OS Security Server RACROUTE Macro Reference*.

IBM makes changes to the templates to add new segments to the RACF database, or to add new fields to existing segments. The templates include level information that allows RACF to compare two sets of templates and determine which has the later set of definitions. The level information consists of a 7–character FMID or APAR level, an 8–digit release level, and an 8–digit APAR level. Each new RACF release increments the release level, and each APAR that ships templates increases the APAR level. RACF initialization and the IRRMIN00 utility use this level information to determine the relationship between the different copies of the templates on the system.

There are three copies of the templates for a system:

- The latest version shipped with RACF is in the CSECT IRRTEMP2. The CSECT resides in the load modules for both RACF initialization and the IRRMIN00 utility.
- The RACF database contains a copy of the templates. This copy of the templates controls how programs that access the database directly, such as the RACF database unload utility and IRRUT200, process database records. The IRRMIN00 utility, when PARM=NEW or PARM=UPDATE is specified, writes the templates to the database from IRRTEMP2.
- There is an in-storage copy, which RACF commands and processes other than utilities use. This copy controls how users and programs that access the database through RACF process database records. RACF initialization builds the in-storage copy at IPL time. You can replace the in-storage copy by running the IRRMIN00 utility with PARM=ACTIVATE specified. You can display the level of the templates with the SET LIST command. For information on the SET LIST command, see “Listing the attributes of the local node” on page 154.

You should ensure that all three copies are at the same level.

Keeping all copies of your database templates at the same level

Any time you install a new release of RACF, or install a PTF that includes new templates, you need to insure that all three copies of the templates are at the same level. You use IRRMIN00 to do this. For a detailed description of the IRRMIN00 utility, see “RACF database initialization utility program (IRRMIN00)” on page 208.

There are three cases to consider:

- You install a new release of RACF, or a PTF that includes new templates and requires an IPL, and you remember to run IRRMIN00 PARM=UPDATE before you do the required re-IPL.
- You install a new release of RACF, or a PTF that includes new templates and requires an IPL, and you re-IPL without running IRRMIN00 PARM=UPDATE.
- You install a PTF that has new templates and does not require an IPL.

Steps for synchronizing the database templates when you install a new release of RACF, or a PTF that includes new templates and requires an IPL, and you have not yet re-IPLed: A PTF might require an IPL if it includes new function and some of the changed modules reside in LPA.

Before you begin: You need to have installed the new release or PTF, but not re-IPLed. If you have already re-IPLed, see "Steps for synchronizing the database templates when you install a new release of RACF, or a PTF that includes new templates and requires an IPL, and you have re-IPLed without running IRRMIN00."

Perform the following steps to ensure that all copies of the database templates are at the same level.

1. Run IRRMIN00 specifying PARM=UPDATE to update the templates on the database from IRRTEMP2.

   Note: If you want to create a *new* RACF database, specify PARM=NEW instead of PARM=UPDATE. Be aware that this effectively erases all the profiles on the database.

---

2. IPL the system.

   RACF initialization determines that the latest level of the templates is already on the database, and builds the in-storage templates from the database version.

When you are done, both the in-storage templates and those on the database are the latest level, the level shipped in IRRTEMP2. You do not need to run IRRMIN00 with PARM=ACTIVATE.

Steps for synchronizing the database templates when you install a new release of RACF, or a PTF that includes new templates and requires an IPL, and you have re-IPLed without running IRRMIN00:

Before you begin: You need to have installed the new release or PTF, and you have already done the required re-IPL without running IRRMIN00.

If you re-IPL after installing the new release or PTF, without running IRRMIN00 PARM=UPDATE, RACF initialization determines that the level of the templates in IRRTEMP2 is higher than the level of the templates on the database. It builds the in-storage templates from the latest level available, the level in IRRTEMP2. As a result, the in-storage templates and the templates on the database are not at the same level, and RACF issues message ICH579E to inform you of that. An example of that message is:

```
ICH579E

RACF TEMPLATES ON DATABASE ARE DOWNLEVEL: HRF7708
00000000.00000000; USING TEMPLATES AT LEVEL HRF7708
00000010.00000000 FROM IRRTEMP2. RUN IRRMIN00 PARM=UPDATE.
```

For a description of message ICH579E, see *z/OS Security Server RACF Messages and Codes*. RACF initialization completes successfully. You do not need to re-IPL. Run IRRMIN00 specifying PARM=UPDATE to write the templates from IRRTEMP2 to the database so that utilities that use the database templates rather than the in-storage templates can process correctly.

Until you run IRRMIN00 with PARM=UPDATE, you might get error messages from IRRUT200 or BLKUPD during some operations, and the RACF database unload

utility will not unload new fields. Also, products that read the database directly and process the database template blocks will have problems with profile information related to new or updated templates.

Perform the following step to update the database with the latest templates in IRRTEMP2.

1. Run IRRMIN00 specifying PARM=UPDATE.

When you are done, both the in-storage templates and those on the database are the latest level, the level shipped in IRRTEMP2. You do not need to run IRRMIN00 with PARM=ACTIVATE.

Steps for synchronizing the database templates when you install a PTF that has new templates and does not require an IPL: A PTF does not require an IPL if it includes new function and all of the changed modules reside in LINKLIB.

Before you begin: You need to have installed the PTF.

Perform the following steps to ensure that all copies of the database templates are at the same level.

1. Run IRRMIN00 specifying PARM=UPDATE.

   IRRMIN00 writes the new templates from IRRTEMP2 to the database.

---

2. Run IRRMIN00 specifying PARM=ACTIVATE.

   IRRMIN00 replaces the existing in-storage templates with the ones on the database. You do not need to re-IPL.

   The SYS1.SAMPLIB member ACTRDS illustrates how to invoke IRRMIN00 specifying PARM=ACTIVATE.

---

3. If the PTF updated the dynamic parse dataset (IRRDPSDS), invoke the IRRDPI00 command specifying UPDATE.

   You need to ensure that the dynamic parse dataset and the in-storage templates are at the same level. Most installations invoke IRRDPI00 automatically during an IPL to accomplish that. Because you don't need to re-IPL, you need to invoke IRRDPI00 yourself. The SYS1.SAMPLIB member ACTRDS illustrates how to invoke IRRDPI00 specifying UPDATE. For more information on dynamic parse, see "Dynamic parse" on page 64.

When you are done, both the in-storage templates and those on the database are the latest level, the level shipped in IRRTEMP2. The dynamic parse data set is at the latest level.

Multiple data set support

You can maintain all of your RACF profiles in one data set or divide your RACF database between multiple data sets. The data sets can be on different devices. Using data sets on different devices has these benefits:

- It can improve performance by reducing device contention.
- It can improve reliability, because if one device experiences an I/O failure, the others might be unaffected.

A RACF database can have as many as 90 data sets.

Backup RACF database

RACF allows you to provide a backup database to which you can switch without a re-IPL should your primary RACF database fail. A backup RACF database reflects the contents of the primary database. Once the installation has created the backup database, RACF can maintain it automatically.

You can decide to back up all of the data sets in your primary database, or some of them, depending on the needs of your installation. Use the RACF data set name table (ICHRDSNT) to control the amount of updating to the backup database. For information on ICHRDSNT, see "The data set name table" on page 37.

RACF allocates the data sets for the backup database at the same time it allocates the data sets for the primary database; therefore, the backup data sets must be online during RACF initialization. In case of a failure on a data set in the primary database, RACF enables you to switch to your backup RACF database without having to re-IPL. The primary and backup databases should reside on different real devices and on different paths. For a discussion of how to handle database failures, see "Failures on the RACF Database" on page 309.

For additional information on backing up your database, see "Creating backup RACF databases" on page 17.

Taking additional backup measures

In addition to creating a backup database, you should periodically take dumps of the RACF database. The dumps could be part of the procedure to create backup copies of other important system data. To make the copies usable, you should create them with a dump/restore program while the system is inactive. If an inactive system cannot be guaranteed, you should use IRRUT200, which issues the proper serialization for the RACF database. IRRUT200 can produce only disk output; for tape, you should provide an additional copy step. Of course, the RACF databases and all copies should be RACF-protected at all times.

Shared RACF databases

Your RACF database can be shared by any combination of the following systems:

- MVS running native
- MVS running as a guest machine of VM
- VM running native
- VM running as a guest machine of VM

Note: In a remote sharing environment, a system configured as an RRSF node can share a RACF database with a non-RRSF system, including a system running VM or running an MVS level lower than MVS/ESA™ 4.3. Be aware that if the RRSF node is using password synchronization or automatic direction, database changes made from a down-level system will not be propagated to other RRSF nodes, and database inconsistencies can result. See Chapter 5, "RACF remote sharing facility (RRSF)," on page 119 for more information.

When the RACF database is to be shared, the device on which the database resides must be configured as shared, or damage to the database is likely. Both primary and backup databases must be shared. For information on how to configure a device as shared, see *z/OS HCD User's Guide*.

The RACF database must match the latest level of code on the sharing systems. The IRRMIN00 utility is used to update the database when a new release or service level is available. The database is downwardly compatible. For example, if an OS/390 Release 4 Security Server system is sharing a RACF database with a RACF 2.2 system, the database must be at the OS/390 Release 4 level, but the RACF 2.2 system can successfully use it. Systems running the Security Server are downwardly compatible with earlier releases of the Security Server, and are downwardly compatible with RACF Version 2 releases.

If you are sharing a RACF database between systems at different levels, you can run any of the utilities, except IRRMIN00, from any of the sharing systems. For example, if an OS/390 Version 2 Release 10 system is sharing a database with an OS/390 Version 2 Release 6 system, you can run IRRUT400 from either OS/390 system. If an OS/390 Version 2 Release 8 system is sharing a database with a z/OS Version 1 Release 2 system, you can run IRRUT400 from either the OS/390 system or the z/OS system.

Attention: If you have run the IRRIRA00 utility to convert the RACF database to application identity mapping stage 1 or later, you should not use the IRRUT400 utility from a downlevel system.

If you are sharing a database between VM and MVS, we recommend that you run the utilities from the MVS side for ease-of-use, recovery, and error-reporting reasons.

General considerations

You must use the same password authentication algorithm on all systems sharing the database. For example, if you use the Data Encryption Standard (DES) algorithm, you must use it on all systems sharing the database. The DES algorithm is the default password authentication algorithm beginning with RACF 2.1, but the masking algorithm is the default for RACF 1.9 and 1.9.2. For more information on password authentication, see “Password authentication options” on page 54.

All sharing systems must use the same data set names. Ensure that the data set name table (ICHRDSNT) on each system uses the same data set names.

If you have split your database, the database range table (ICHRRNG) must be the same on all systems.

All sharing systems must have compatible class descriptor tables (ICHRRCDE).

Considerations for the SCICSTST and UCICSTST classes

Beginning with OS/390 Version 2 Release 8 and in Version 2 Releases 6 and 7 with PTF UW90545 installed, the maximum length of profile names and member names for the SCICSTST and UCICSTST classes increased from 17 to 25 characters. If a system with the changed class definitions shares a RACF database with a lower-level system that does not have the changed definitions, the following considerations apply:

- Do not define profiles or members using the longer names if you have made the classes active via the SETROPTS CLASSACT command.
- Do not use RACF remote sharing to manually or automatically direct a command to a system that does not have the changed definitions if the command references an SCICSTST or UCICSTST profile or member with a name longer than 17 characters. The command would run on the local system but would fail on the target system.

- Do not have the RACGLIST function active for either the SCICSTST or UCICSTST class. RACGLIST shares RACLISTed profiles between classes that share the database, and the RACLISTed profiles for these classes would not be compatible across the systems.

RACGLIST considerations

The RACGLIST class allows the security administrator to specify selected classes for special processing during SETROPTS RACLIST and RACROUTE REQUEST=LIST,GLOBAL=YES processing. The RACGLIST class should be used only if all systems sharing the RACF database are in the same GRS complex. The major name SYSZRAC2 cannot be in the exclusion resource name list (RNL).

For information about using the RACGLIST class for the SCICSTST or UCICSTST classes, see "Considerations for the SCICSTST and UCICSTST classes" on page 9.

RACF sysplex communication

When multiple systems share the RACF database, two additional options are available:

- The **RACF sysplex communication option** facilitates system administration.
- Systems enabled for sysplex communication can use the **RACF data sharing option** if all the systems are running MVS 5.1 or above and a coupling facility is available. RACF data sharing might improve data access performance.

For more information on these options, see "Sysplex considerations" on page 84.

Sharing RACF data without sharing a database

Installations often find it very useful to share RACF data between systems. However, in order for systems to share a RACF database they must be in close enough physical proximity to physically share the device on which the database resides. The RACF remote sharing facility (RRSF) expands an installation's ability to share RACF data by removing the restrictions of shared DASD. It allows you to configure your systems into a network of *RRSF nodes* communicating via VTAM and APPC/MVS, and share RACF data between these nodes regardless of their physical proximity. You can:

- Give each RRSF node its own copy of the same RACF database, and use remote sharing functions to keep the databases synchronized. Or, selectively synchronize subsets of the database information, such as the user profiles.
- Administer RACF databases remotely–authorized users logged on to one system can direct a RACF command to run on one or more other systems in the RRSF network.
- Automatically synchronize passwords for specified user IDs on systems in the RRSF network.

For more information on the RACF remote sharing facility, see Chapter 5, "RACF remote sharing facility (RRSF)," on page 119.

Creating a RACF database

To create a RACF database you must allocate it, catalog it, and use IRRMIN00 with PARM=NEW to format it. For information on using IRRMIN00, see "RACF database initialization utility program (IRRMIN00)" on page 208. A RACF database must be allocated in one extent.

Recommendation: Make a RACF database unmovable. If an active database is moved from where RACF thinks it is, for example, by a DFSMSdss™ DEFRAG operation on the volume, results are unpredictable. Requests for RACF services might fail, and profile updates might be lost. If you choose to make a RACF database movable, you should put procedural controls in place that guarantee that the RACF database is not moved unless an RVARY INACTIVE command is issued.

SYS1.SAMPLIB member RACJCL provides sample jobs to allocate, catalog and format a RACF database. The samples can be modified to fit your installation's requirements. The following is a sample job which creates a RACF database:

```
//INITRDS  JOB  ,'INITIALIZE NEW DS',
//            MSGLEVEL=(1,1),TYPRUN=HOLD
//INITALZE EXEC PGM=IRRMIN00,PARM=NEW
//STEPLIB  DD  DSN=SYS1.LINKLIB,DISP=SHR,   **MUST BE LIBRARY WITH **
//             UNIT=YYYY,VOL=SER=YYYYYY     **NEW RELEASE IRRMIN00 **
//SYSPRINT DD  SYSOUT=*
//SYSRACF  DD  DSN=SYS1.RACF,DISP=(NEW,CATLG),
//             UNIT=XXXX,VOL=SER=XXXXXX,
//             SPACE=(CYL,(XX),,CONTIG),
//             DCB=DSORG=PSU
/*
```

Note: If you include a SYSTEMP DD statement in your JCL, it is ignored.

The CONTIG statement in the example ensures that the database is allocated as a single extent. The DCB=DSORG=PSU makes the database unmovable. When the database is on an SMS-managed volume, you cannot specify PSU. Instead, you must specify PS, and be sure to exclude the RACF database from any DEFRAG-type operation explicitly, via control statements.

See "RACF Database Storage Requirements" on page 333 for information on estimating the amount of space a database will require.

Finding a location for the RACF database

The data sets in a RACF database can reside on any DASD device that is supported by the operating system. Each volume containing a RACF database data set should be permanently resident. If RACF is heavily used and you elect to use a single data set for the RACF database, plan to put the data set on a device accessed by the channel and control unit least likely to impact system performance.

Each data set in a RACF database must be a contiguous, single-extent, non-VSAM data set that resides on a DASD volume, and it must be cataloged. When you IPL the system, RACF allocates and opens the data set, and MVS updates RACF's control blocks with the physical location of the data set on the volume.

If you need to move your RACF database, copy it to the new location following the procedures in "Copying your database."

Copying your database

If you have to copy your database, you do not need to re-IPL the system.

The following sample procedures assume that you have primary and backup databases.

When you issue the following RVARY commands, name the data sets in the database using the DATASET operand, to avoid accidentally processing the wrong data sets. Do not let the data set names default.

Note: If your database has multiple data sets and you're using IRRUT200, you need to follow the procedures below for each data set individually. If you're using IRRUT400, you can process multiple data sets all at once.

Procedure for the primary database

1. Ensure that the backup database is active (RVARY LIST, followed, if necessary, by RVARY ACTIVE).
2. Issue RVARY SWITCH. The original primary is now an inactive backup.
3. Copy the current primary database (the original backup) using IRRUT200, or IRRUT400 with LOCKINPUT.

   Uncatalog the current backup database (the original primary). Ensure that the newly-created database that you want to use as the new primary database has been cataloged, and ensure that it has the same name as the original primary database. This must be done on all systems that share the RACF database.
4. Issue RVARY ACTIVE for the new primary database (the current backup).
5. If you used IRRUT400 with LOCKINPUT in step 3, run IRRUT400 with UNLOCKINPUT to unlock your current primary (the original backup).
6. Issue RVARY SWITCH (this inactivates the original backup).
7. Issue RVARY ACTIVE for the original backup.

Note: After an RVARY SWITCH when your backup is inactive, your primary and backup databases may become out of synch. If this is a concern to you, the safest approach is to use IRRUT400 with LOCKINPUT to perform the copy. But note that even in this scenario, a window exists between steps 6 and 7 where your databases may become out of synch. The recommended scenario is to use IRRUT200 to do the copy at a time when no profiles are being updated in your RACF database. This gives you the fastest copy performance.

Procedure for the backup database

1. Ensure that the backup database is inactive and deallocated. To determine the status of the backup database, use the RVARY LIST command.
2. The next step depends on the status of your backup database.
   - If it is active, issue RVARY INACTIVE for the backup.
   - If it is inactive and allocated, issue RVARY ACTIVE for the backup, and then RVARY INACTIVE.
   - If it is inactive and deallocated, go to the next step.
3. You have two options for this step. Use step 3a when you know your database is not being updated, or you are willing to risk losing database updates in return for a quicker copy. Use step 3b on page 13 when you want to insure that no database updates are lost, even if it takes a little longer.
   a. Copy the backup database using IRRUT200 or IRRUT400 (without LOCKINPUT) and uncatalog the database. For information on determining which utility to use, see Chapter 7, "RACF database utilities," on page 201.

      Ensure that the newly-created database that you want to use as the backup database is cataloged, and ensure that it has the same name as the original, backup database.

b. Copy the primary database using IRRUT400 with LOCKINPUT to create a new backup. Uncatalog the original backup.

4. Issue RVARY ACTIVE for the new backup database.
5. If you did step 3b, run IRRUT400 with UNLOCKINPUT to unlock your primary database.

Using DFSMSdss DEFRAG

If you choose to run this program to compact a volume on which a RACF database resides, there are several ways of doing so:

- Designating the RACF data sets unmovable indicates that all the data sets for the RACF database are unmovable. Run DFSMSdss DEFRAG, which will compress the rest of the data on the volume, but will not alter the location of the RACF data sets.
- Issue RVARY, with the SWITCH or INACTIVE operand, to deactivate and deallocate the databases. Do this when other users or jobs are not on the system, because they may experience failures while the database is inactive. Next, run the DFSMSdss DEFRAG program to compress the volume. Last, reissue the RVARY command with either the SWITCH or ACTIVE operand to automatically reactivate and reallocate the databases. When you issue RVARY SWITCH or ACTIVE, the RACF control blocks that describe the physical location of the database are rebuilt.

  For a more detailed description of the RVARY command, see *z/OS Security Server RACF Command Language Reference* and "The RVARY Command" on page 305.
- If you cannot make the RACF database unmovable, and you want to run DEFRAG with the database active, you must use the EXCLUDE operand on the DFSMSdss control statements to explicitly exclude the RACF database from being moved during the DEFRAG operation.

DFSMS enhanced data integrity (EDI)

DFSMS provides an enhanced data integrity function for physical sequential (PS) data sets that you can activate using a SYS1.PARMLIB member. You can also exempt selected data sets from enhanced data integrity processing. When enhanced data integrity is active, if someone allocates a non-exempted physical sequential data set using DISP=SHR, and attempts to OPEN the data set for output, if the data set is already open for output DFSMS detects an error and abends the second OPEN attempt.

The RACF database is allocated as a physical sequential data set. RACF processing ensures that using the RACF database does not cause abends due to enhanced data integrity processing. You do not need to exempt the RACF database from enhanced data integrity processing.

Monitoring the usable space in your RACF database

Over time, the usable space in a data set in your RACF database decreases in two ways:

- As new profiles are added to the data set, the amount of available space decreases.
- As existing profiles are updated, the available space might become fragmented. When an existing profile is updated, there might not be enough room to update it in its current position in the data set, and it might have to be rewritten in a larger contiguous slot. Therefore, a data set that appears to have plenty of available

space might be so fragmented that an update to a profile fails due to insufficient space because there is not a large enough contiguous slot to accommodate it.

In order to anticipate and prevent running out of usable space in your RACF database, periodically run the RACF database verification utility, IRRUT200, against each data set in the database, to check on the amount of available space and its degree of fragmentation. The MAP ALL function of IRRUT200 reports the percentage of the data set that is in use and produces an encoded map of the BAM (block availability mask) blocks that is useful in determining if the data set has become fragmented. For more information on running IRRUT200, see “RACF database verification utility program (IRRUT200)” on page 218.

When you determine that your usable space is running low, run the RACF database split/merge/extend utility program, IRRUT400, to copy your data set to a larger data set, or, if fragmentation is the only problem, to another data set the same size. As IRRUT400 copies the data set, it rebuilds it, undoing any fragmentation that has occurred. For a procedure to follow to do the copy, see “Copying your database” on page 11 and follow the instructions there using IRRUT400. For information about IRRUT400, see “RACF database split/merge/extend utility program (IRRUT400)” on page 232.

Chapter 2. Performance considerations

The RACF database . . . 16
Selection of control unit and device . . . 16
Shared RACF database . . . 16
RACF remote sharing facility . . . 16
RACF sysplex data sharing . . . 16
Multiple data sets . . . 17
RACF sysplex data sharing . . . 17
RACF remote sharing facility . . . 17
Database housekeeping . . . 17
Creating backup RACF databases . . . 17
Options for updating backup databases . . . 18
Resident data blocks . . . 19
RVARY SWITCH command . . . 19
Auditing . . . 20
Operands requiring the AUDITOR attribute . . . 20
APPLAUDIT . . . 20
AUDIT . . . 20
CMDVIOL . . . 20
LOGOPTIONS . . . 21
OPERAUDIT . . . 21
SAUDIT . . . 21
SECLABELAUDIT . . . 21
SECLEVELAUDIT . . . 21
RACF commands . . . 21
RACF utility programs . . . 22
BLKUPD . . . 22
IRRUT200 . . . 22
Failsoft processing . . . 22
Erase-on-scratch . . . 23
Installation-written exit routines . . . 24
Using global access checking . . . 24
The SETROPTS command . . . 24
Using SETROPTS RACLIST and SETROPTS GENLIST . . . 25
RACLIST processing . . . 25
Refreshing SETROPTS RACLIST processing . . . 27
GENLIST processing . . . 28
Refreshing in-storage generic profiles . . . 28
Using SETROPTS INITSTATS and SETROPTS STATISTICS . . . 29
INITSTATS processing . . . 29
STATISTICS processing . . . 30
Identification, verification, and authorization of user IDs . . . 31
User identification and verification . . . 31
RACROUTE REQUEST=VERIFY or VERIFYX processing . . . 31
RACROUTE REQUEST=SIGNON processing . . . 31
Improving verification performance using VLF . . . 32
RACROUTE REQUEST=AUTH processing . . . 32
RACROUTE REQUEST=FASTAUTH processing . . . 33
Using generic profiles . . . 33
Mapping UIDs to user IDs and GIDs to group names . . . 34
z/OS UNIX System Services applications . . . 34
Large profiles . . . 34
Large groups . . . 35

© Copyright IBM Corp. 1994, 2004

Universal groups 35

The effect that RACF has on system performance depends directly on the type and number of RACF functions performed. The system programmer has direct control over some of these functions; this chapter identifies areas where performance issues should be addressed.

Ordinarily, when the RACF database is shared by multiple systems, RACF uses the hardware RESERVE/RELEASE capability to serialize access to the database. Using global resource serialization (GRS) can minimize problems sometimes associated with hardware RESERVEs. An installation can explicitly convert hardware RESERVEs to GRS ENQs.

Note: When enabled for sysplex data sharing and operating in data sharing or read-only mode, RACF always uses GRS ENQs rather than RESERVEs.

The RACF database

There are several decisions to make concerning your RACF database. Performance is directly affected by I/O contention. System performance can be affected in the following ways:

Selection of control unit and device

The choice of the DASD device and control unit for the RACF database can affect RACF and system performance in a high-use environment. For example, device saturation occurs earlier for an IBM 3350 than it does for an IBM 3380.

Do not place the RACF database on the same control unit or device as other frequently used data sets. Placing it on such a device degrades both system and RACF performance.

If the device activity, because of RACF database I/O, warrants it, consider placing the RACF database behind a cached control unit such as the IBM 3990-3.

Shared RACF database

RACF is designed so that its database can be shared between processor complexes while data integrity is maintained. Because of the need to serialize RACF database processing, there might be some I/O contention. However, if your installation does not share the database, you optimize performance if you place the RACF database on a non-shared device. See also “Using the global resource serialization function” on page 80.

RACF remote sharing facility

If you have multiple systems sharing a RACF database and contention is a problem, you can make a copy of the database for each of the sharing systems and use the RACF remote sharing facility to keep the databases synchronized. Performance should improve because contention is reduced on each of the databases. See Chapter 5, “RACF remote sharing facility (RRSF),” on page 119 for more information.

RACF sysplex data sharing

The RACF sysplex data sharing option is available for systems running RACF 2.1 or later on MVS 5.1 or later. RACF sysplex data sharing is an optional function that can:

- Facilitate system administration

- Provide consistent sysplex-wide security
- Improve performance in some environments

Operating RACF in data sharing mode (available only when RACF is enabled for sysplex communication) requires use of a coupling facility.

RACF sysplex data sharing is designed to address problems that can occur when many systems share a RACF database. RACF uses the coupling facility as a large sysplex-wide store-through cache for the RACF database that reduces contention and I/O to DASD. Serialization is done using GRS instead of RESERVE/RELEASE while in data sharing mode. RACF sysplex data sharing also provides improved efficiency of invalidating resident data blocks. For more information on sysplex data sharing, see “Sysplex considerations” on page 84.

If you have a non-shared database, you can still take advantage of the possible performance improvement that the coupling facility offers. See “Using the coupling facility with a single MVS image” on page 89 for details.

Multiple data sets

If you split the RACF database between multiple data sets, you can reduce the effect of I/O on performance, because:

- Each data set might receive fewer requests
- Each data set might be smaller, in which case each request requires fewer I/O requests and fewer movements of the activator arm on the device
- Each data set can have up to 255 resident-data blocks, optimizing I/O and increasing the amount of in-storage data. See also “Resident data blocks” on page 19.

RACF sysplex data sharing

If you have previously split your RACF database to get better performance, you might find that using the coupling facility and data sharing mode causes performance to improve. Because of the improved performance, you might decide that you can recombine your databases. When you are using data sharing mode with your split configuration, examination of the RACF database I/O rate and the coupling facility statistics should help you evaluate this possibility.

RACF remote sharing facility

For performance problems caused by the aggregate database I/O rate for multiple systems, an alternative to splitting the database is to use the RACF remote sharing facility to provide a separate copy of the database for each system or group of systems.

Database housekeeping

Reorganizing the database using IRRUT400 keeps related data clustered together and reduces I/O.

Creating backup RACF databases

If you have active backup RACF databases, you increase the amount of processing performed for updates to RACF databases. However, you also reduce the amount of time it takes to recover if a database error occurs.

To create a backup RACF database, copy the current RACF database and specify the new database configuration and backup options to RACF. To keep your backup database identical with your primary database, do not make any further updates to

the primary database before you activate the backup database. Create and activate the backup database when no other users or jobs are active in the system.

There are two utilities you can use to create a backup database:

- IRRUT200 serializes on the RACF database and creates an exact, block-by-block copy of it.

  This exact copy can help performance when you are maintaining statistics on your backup database. IRRUT200 can be used only if you are creating a backup database that is the same size and on the same device type as the input database. For more information, see “RACF database verification utility program (IRRUT200)” on page 218.
- IRRUT400 creates a copy of your database and can be used to change its size.

  IRRUT400 also reorganizes the contents of the output RACF database. Use this utility if you are copying between different device types. You can also use IRRUT400 to extend the RACF database before it becomes full. For important information, see “RACF database split/merge/extend utility program (IRRUT400)” on page 232.

It is important to use the RACF-provided utilities when copying an active RACF database, because they serialize to protect the data in your database. If, however, your database is inactive, you can use other block copy utilities, such as IEBGENER.

Options for updating backup databases

The RACF data set name table (ICHRDSNT) specifies the data set names for both the primary and backup RACF databases, and the recovery option. If the primary database is split, you specify several pairs of entries. If you elect to use the RACF data set name table (ICHRDSNT), you can choose from three backup options:

1. **All updates duplicated on the backup database**

   When you update the primary database, the backup database is also updated. If you choose this option, your backup database must be a copy of the primary database that existed at RACF initialization. Switching to this backup database is transparent to the users.

   The cost, in terms of RACF processing for this option, is high if you use many discrete profiles and do not use SETROPTS RACLIST processing.
2. **All updates, except for statistics, duplicated on the backup database**

   This option is similar to the first option, except that changes you make to the primary database for the sole purpose of updating statistics are not made on the corresponding backup database. If you are maintaining statistics on the primary database and you must switch to the backup database, you might lose some statistics.

   Note: However, if SETROPTS INITSTATS is on, a limited subset of statistics is maintained on the backup.

   The cost, in terms of RACF processing for this option, can be appreciable if a high proportion of your activity is changing RACF profiles. However, the overhead is lower than for the first option, and your backup database is current in the event of an error on your primary.

   We recommend that you use this option in your data set name table.
3. **No updates duplicated on the backup database**

With this option, your backup database is allocated but inactive. When you make changes to the primary database, the corresponding backup database is not updated. If you switch to this backup database when there is a failure in your primary database, you bring a down-level RACF database into operation.

Note: If you activate the backup database, RACF will start recording the updates on the backup.

The cost, in terms of RACF processing for this option, is negligible, but system operation and recovery could be difficult, depending on how out-of-date the information in the database is.

For more information, see "The data set name table" on page 37.

Resident data blocks

RACF enables you to request common storage area buffers to reduce the database I/O. If you have a data set name table (ICHRDSNT), you can specify the number of resident data blocks for each data set in the primary RACF database.

If RACF is enabled for sysplex communication, note the following:

- An installation-defined data set name table is required. Each member of the sysplex data sharing group should specify the same data set names, in the same order, in the data set name table. However, if a system doesn't specify the same names or order, RACF ignores the names or order specified by that system and uses the data set names and order already in use by the first system that IPLed enabled for sysplex communication.
- Buffers are allocated for backup (as well as primary) database data sets. RACF calculates the buffer size for each backup data set as 20 percent of the size of the corresponding primary data set buffer.

For each data set in the primary RACF database, an installation can assign from 0 to 255 resident data blocks. If you do not have a data set name table (ICHRDSNT), the default is 10 resident data blocks. If you have a data set name table, RACF uses the number specified in the table, with a minimum of 50 when RACF is enabled for sysplex communication. For best performance, specify as large a number of buffers as you can afford, preferably 255. The storage is in ECSA.

You can have resident data blocks when the RACF database resides on a shared device. RACF updates the resident data blocks to ensure that all processors use the latest level of the blocks. When resident data blocks are used for a shared RACF database, some resource statistics might not be updated. For more information, see "Selecting the number of resident data blocks" on page 41.

RVARY SWITCH command

When you issue RVARY SWITCH, RACF associates a set of buffers with the new primary database (the old backup database) and dissociates the buffers from the old primary database (the new backup database).

If RACF is enabled for sysplex communication when you issue RVARY SWITCH, RACF associates the larger buffer with the new primary database (the original backup database) and the smaller buffer with the new backup database (the original primary database).

Auditing

An installation can control the amount of auditing done on its system by activating various RACF options.

The more auditing performed, the more system performance is negatively affected. Auditing of frequent events affects performance more than auditing occasional ones.

When auditing is requested, RACF produces system management facility (SMF) records to log the detected accesses and attempts to access RACF-protected resources. The more auditing done, the larger the SMF files that must be analyzed by the system auditor.

System-wide auditing options can be controlled by users with the AUDITOR attribute. However, users who have the SPECIAL or group-SPECIAL attribute, or who are the owners of a resource profile, *are* allowed to specify the AUDIT operand on the ADDSD, ALTDSD, RALTER or RDEFINE commands.

Operands requiring the AUDITOR attribute

Users with the AUDITOR attribute can specify the GLOBALAUDIT operand on the ALTDSD or RALTER command. GLOBALAUDIT enables the auditor to log events in addition to those chosen by the owner of the profile.

Users with the AUDITOR attribute can specify the UAUDIT operand on the ALTUSER command. UAUDIT enables the auditor to log all RACF-related activities for a specific user.

Users with the AUDITOR attribute can specify the following operands of the SETROPTS command:

- APPLAUDIT or NOAPPLAUDIT
- AUDIT or NOAUDIT
- CMDVIOL or NOCMDVIOL
- LOGOPTIONS
- OPERAUDIT or NOOPERAUDIT
- SAUDIT or NOSAUDIT
- SECLABELAUDIT or NOSECLABELAUDIT
- SECLEVELAUDIT or NOSECLEVELAUDIT

APPLAUDIT

If APPLAUDIT is specified, and if AUDIT is specified for the APPL profile associated with APPC/MVS, user verification during APPC/MVS transactions is audited.

Persistent verification (PV) support directly affects the amount of auditing. Without PV support, two SMF records per APPC transaction are produced. With PV support, two SMF records per user are produced: one at signon and one at signoff. The number of SMF records created is reduced.

AUDIT

Causes an SMF record to be produced when RACF profiles are changed by a RACF command for a specified class and when a RACROUTE REQUEST=DEFINE is issued (whether or not a profile is changed). If you specify AUDIT for the DATASET class, an SMF record is produced for every data set created or deleted.

CMDVIOL

Used to log violations detected during RACF command processing.

LOGOPTIONS
Enables an installation to control logging on a class, as opposed to a profile basis.

Note: Choosing ALWAYS (always log) for frequently used classes quickly degrades your system's performance.

OPERAUDIT
Used to audit all accesses to resources granted because the user has the OPERATIONS attribute or the group-OPERATIONS authority.

SAUDIT
Used to log the commands issued by users with the SPECIAL or group-SPECIAL attribute.

SECLABELAUDIT
Used to audit access attempts in the SECLABEL class, for RACF-protected resources.

SECLEVELAUDIT
Used to audit access attempts to the specified installation-defined security level, for RACF-protected resources.

For more information on the command operands, see *z/OS Security Server RACF Command Language Reference*. For more information on activities that are never logged, optionally logged, and always logged, see *z/OS Security Server RACF Auditor's Guide*.

RACF commands

RACF commands that read or process a large number of profiles (for example, SEARCH, LISTDSD with the ID or PREFIX operands, LISTGRP *, LISTUSER *, and RLIST *classname* *), can cause contention for the RACF database. If RACF is not enabled for sysplex communication (or if RACF is enabled for sysplex communication but the system is running in non–data sharing mode), RACF serializes access to the database with RESERVE/RELEASE for each profile that it processes. When running in data sharing or read-only mode, RACF uses ENQs to serialize database access.

Depending on the amount of contention, the processing of other RACF commands might be slowed down, and systems sharing the RACF database might appear to be locked out. This contention might be reduced if the resident data-block option is in effect, and if the data can be located in one of the blocks. For more information, see "Using the global resource serialization function" on page 80. If RACF is in data sharing mode, contention is reduced if the data is found in the coupling facility.

If you are saving ACEEs by using VLF, issuing commands that change user profile information in the database might degrade system performance. For more information, see "Removing information from VLF" on page 68.

To reduce the impact on the system, use slack times to issue RACF commands that perform large-scale operations against the RACF database. You can also use the database unload utility (IRRDBU00) to obtain information from a copy of the RACF database. For information on IRRDBU00 see *z/OS Security Server RACF Security Administrator's Guide*.

RACF utility programs

When one of the RACF utility programs is processing a data set in the RACF database, that data set might be unavailable for other use (such as authorization checking). To reduce the impact on the system and on RACF performance, it is recommended that you run the RACF utility programs during slack time in system operation.

You can also reduce the impact to your system by unloading your RACF database. The output from the database unload utility (IRRDBU00) can be used to collect information in the RACF database. For information on IRRDBU00, see *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF Macros and Interfaces*.

BLKUPD

The BLKUPD command can result in RESERVEs (or ENQs if RACF is running in data sharing or read-only mode) being held against the RACF database from the time the terminal user issues the READ subcommand until the user issues the corresponding END command.

IRRUT200

The RACF database-verification utility program, IRRUT200, can create a working copy of the RACF database by copying to the database defined by the SYSUT1 DD statement. If you use the copy function, IRRUT200 performs verification on the copied database, and the input RACF database is available during subsequent processing.

For more information on the utilities, see Chapter 7, "RACF database utilities," on page 201.

Failsoft processing

Failsoft processing occurs when no data sets in the primary RACF database are available (RACF is installed but inactive). Failsoft processing degrades system performance and system security.

There are several reasons why failsoft processing might be in effect on your system:

- RACF is installed but does not know the name of the primary master data set.
- Failures occurred during RACF initialization at IPL time.
- An RVARY INACTIVE command was issued, inactivating all data sets in the primary database.

If RACF is enabled for sysplex communication, failsoft processing can also result when:

- The system is running in sysplex local mode.
- A data set in the RACF database does not reside on a shared device.
- A system is running an MVS release that does not support the RACF data sharing option and attempts to IPL in data sharing mode.
- A system attempting to join an existing IRRXCF00 group is unable to allocate one or more RACF database data sets that are in use by the other systems in the group. The system operator is not prompted for a data set name, and the

system joins the group and begins failsoft processing. A system IPL is required to enable RACF processing when the problem is resolved.
- RACF encountered an internal error while processing a request on behalf of the RACF sysplex data sharing group.

During failsoft processing, the operator is prompted frequently to grant access to data sets. To avoid this situation, we recommend that you have a backup RACF database so that you can issue the RVARY SWITCH command rather than an RVARY INACTIVE command.

If you cannot avoid failsoft processing, limit access to the system and do not run production work. You can also try using the RACROUTE REQUEST=AUTH and RACROUTE REQUEST=DEFINE installation exits to implement failsoft processing in some other way.

For more information on failsoft, see “Failsoft processing” on page 102.

Erase-on-scratch

To physically erase security-sensitive data at the time the data set extents are scratched, RACF and DFSMS provide an erase-on-scratch facility. Erase-on-scratch ensures that when the data set is scratched (deleted or released for reuse), it cannot be read by any program running under control of an IBM operating system. It enables you to protect both single and multivolume DASD data sets.

With the erase-on-scratch facility, you can designate that specific data sets with a particular security level or that all data sets should be physically erased when the data set is deleted or when some of the space that was allocated to the data set is released. During this process, RACF tells DFSMS that data erasure is required.

The erase-on-scratch facility provides a defense against two types of attacks:
- It protects against an attempt to read residual data. This means that no one can allocate a new data set at the same location, open it for input, and read your data. This requires no exotic tools or insider knowledge and can be done quite easily using JCL and an IBM-provided utility such as IEBGENER.
- It defends against an attempt to read data by acquiring physical access to a device and attempting to read its data directly.

Erase-on-scratch might place an additional load on DASDs, which can have an impact on system performance, depending on how much erasure is being performed and how the erasure is being done. However, you can minimize the impact by various means.
- You have the option of controlling which data sets are erased. You can do this when you:
  - Create or modifying a data set profile
  - Delete actual data sets using the RACROUTE REQUEST=AUTH postprocessing exit routines.

  Notes:

  1. If you activate ERASE ALL, an installation exit cannot override the option to prevent data sets from being erased.
  2. Failsoft processing used with erase-on-scratch can affect the erase-on-scratch procedure and overall system performance. If a RACROUTE REQUEST=AUTH with STATUS=ERASE is processed when

RACF is inactive during failsoft processing, and if the preprocessing exit grants authorization checking, the reason code specified in the exit parameter list is passed to the caller of the RACROUTE request. If the reason code equals 0, no erasure is performed. If the exit does not grant authority but failsoft processing does, the reason code from the exit equals 4 and indicates erasure is to be performed.

- Using data erasure with virtual array devices means that the storage subsystem erases data automatically without performance penalty. DFSMS 1.4 with OW29642 checks the erase results from the RVA device. If the data was to be erased, DFSMS checks whether it was erased by the device. If it was not, DFSMS erases the data using other methods.

  Two general ″rules of thumb″ flow from this implementation:

  1. If you are using the DDSR function of IBM's extended data facility product (IXFP), specifying erase-on-scratch has minimal impact because DDSR performs the erasure in the overwhelming majority of cases.
  2. If you have data for which you want to enable erase-on-scratch, allocate the data on DDSR enabled volumes.

By following these two rules, your data can be erased by the storage subsystem in the overwhelming majority of cases. In those rare cases where the storage subsystem was not able to erase the data, DFSMS erases the data using the ERASE CCW. This is also faster than on older devices because it does not need to wait for disk rotation.

For more information, see *z/OS DFSMS: Using Data Sets*.

Installation-written exit routines

Exit routines can add to or reduce the impact on system performance depending on the processing the exit routines perform. For a discussion of the exit routines, see Chapter 8, "RACF installation exits," on page 247.

Using global access checking

You can use global access checking to improve performance of RACF authorization checking for selected resources. Global access checking should be used 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. This can avoid I/O to the RACF database to retrieve a resource profile, and can result in substantial performance improvements.

For more information on the global access checking table, see *z/OS Security Server RACF Security Administrator's Guide*.

The SETROPTS command

Certain operands of the SETROPTS command directly affect system performance:

- RACLIST
- RACLIST REFRESH
- GENLIST
- GENERIC REFRESH
- INITSTATS

- STATISTICS

Using SETROPTS RACLIST and SETROPTS GENLIST

You can optimize performance by carefully deciding whether to use SETROPTS RACLIST or SETROPTS GENLIST for various classes.

The RACLIST operand on the SETROPTS command improves performance by copying generic and discrete profiles for the designated general-resource class, and for each class that can be RACLISTed and that shares the same POSIT value, from the RACF database into a data space.

If you use RACROUTE REQUEST=LIST, you can also improve performance by specifying GLOBAL=YES on the request. GLOBAL=YES stores the REQUEST=LIST results in a data space, which can then be shared by other applications that issue the same request. The additional RACROUTE requests do not access the database, they access the data space built by the first RACROUTE. Additionally, the different applications do not need to individually issue RACROUTE REQUEST=LIST deletes followed by RACROUTE REQUEST=LIST creates to refresh the original RACROUTE. The system administrator can do that by a SETROPTS RACLIST(*classname*) REFRESH, which deletes the existing data space, references the database to rebuild the RACLIST results and stores them in a new data space, which then becomes accessible to any application address space that has issued REQUEST=LIST,GLOBAL=YES for that class.

You can use the RACGLIST class to store the RACROUTE REQUEST=LIST and SETROPTS RACLISTed results on the RACF database. RACGLIST profiles are used during IPL for SETROPTS RACLISTed classes, and when a peer RACF system receives a propagated SETROPTS RACLIST command. (The RACGLIST profiles would also be used on a RACROUTE REQUEST=LIST,GLOBAL=YES if no data space had previously been built on that system.) For a large number of profiles it should be quicker to get the profiles from the RACGLIST class than to read all the individual profiles from the associated classes. It is recommended that you use SETROPTS RACLIST.

The GENLIST operand on the SETROPTS command improves performance by copying generic profiles from the RACF database into storage.

Before issuing a SETROPTS GENLIST or SETROPTS RACLIST for a general resource class, consider:

- Whether you can afford the storage utilization. This applies only to SETROPTS GENLIST.
- Whether you can afford the overhead—an administrator must refresh all profile changes so that they become effective.
- Whether the longer IPL time is acceptable. This applies only to SETROPTS RACLIST, and only if you have a large number of class profiles. Using RACGLIST profiles during IPL might reduce time delays associated with SETROPTS RACLIST.
- Whether the RACGLIST class is active and the class has been defined in the RACGLIST class; additional storage will be required for your database.

You *cannot* use both RACLIST and GENLIST for the same general resource class.

RACLIST processing

The RACLIST operand on the SETROPTS command copies generic and discrete profiles from the RACF database into storage. The profile copies are put in their

own data space. RACF uses these profile copies to check the authorization of any user who wants to access a resource protected by them. Additionally, if the RACGLIST class is active and a profile is defined with the same name as the class being RACLISTed, RACF copies the contents of the data space into *classname*_nnnnn profiles to create the RACLIST data space if the system is IPLed. They are also used on a system which is enabled for sysplex communication by members of a data sharing group that are processing a propagated SETROPTS RACLIST command. They are used to build the RACLIST data space, rather than having each member access the database for each discrete and generic profile in the class being RACLISTed.

Before you use RACLIST, consider how frequently the class is referenced, the number of profiles in the class, and the amount of storage that would be required to hold the profiles. Use SETROPTS RACLIST when the general resource class contains frequently referenced profiles, and global access checking cannot be used (that is, everyone is not allowed access to the resources).

You cannot maintain resource-usage statistics on those profiles for which a SETROPTS RACLIST was issued for the class.

To activate RACLIST processing, a user with the SPECIAL attribute issues the following command:

```
SETROPTS RACLIST(classname...) CLASSACT(classname...)
```

If RACF is enabled for sysplex communication, a SETROPTS RACLIST issued from one system in a sysplex is propagated to the other systems in the data sharing group. If RACF is not enabled for sysplex communication, when you issue a SETROPTS RACLIST on one system, that action is *not* propagated to other systems that share the RACF database; you must issue the command separately for each system, or IPL the other system. See “Shared system considerations” on page 27.

If the following classes supplied by IBM are active, you *must* issue a SETROPTS RACLIST command:

APPCSERV	DEVICES	PROPCNTL	SECLABEL	SYSMVIEW
APPCTP	FIELD	PSFMPL	SERVAUTH	UNIXPRIV
CSFKEYS	NODES	PTKTDATA	STARTED	VTAMAPPL
CSFSERV	OPERCMDS	RACFVARS		

In-storage profiles for the following classes supplied by IBM can be optionally shared by using SETROPTS RACLIST. We recommend that you SETROPTS RACLIST the classes marked with an asterisk (*) if you define profiles in those classes and activate the classes:

ACCTNUM *	DBNFORM	JESINPUT	PERFGRP *	TERMINAL
ALCSAUTH	DCEUUIDS	JESJOBS	PTKTVAL	TMEADMIN
APPCPORT	DIGTCERT *	JESSPOOL	RRSFDATA *	TSOAUTH *
APPCSI	DIGTRING	KEYSMSTR	SDSF	TSOPROC *
APPL	DLFCLASS	LFSCLASS	SERVER	VMBATCH
CBIND	DSNR	LOGSTRM	SMESSAGE	VMCMD
CONSOLE	FACILITY *	MGMTCLAS	SOMDOBJS	VMNODE
CPSMOBJ	FCICSFCT	MQCMDS	STORCLAS	VMSEGMT
CPSMXMP	INFOMAN	MQCONN	SUBSYSNM	WRITER
DASDVOL	JAVA	NETCMDS	SURROGAT	

RACROUTE considerations when using SETROPTS RACLIST: If your application uses RACROUTE REQUEST=AUTH for authorization checking, profiles that were brought into storage with the SETROPTS RACLIST command are accessible.

If your application is an authorized program, it can use RACROUTE REQUEST=FASTAUTH for profiles that are SETROPTS RACLISTed. If your application does not run authorized, it can use RACROUTE REQUEST=FASTAUTH only for profiles brought into storage by a RACROUTE REQUEST=LIST.

If your application uses RACROUTE REQUEST=LIST,GLOBAL=NO for a class, RACF uses locally RACLISTed profiles for authorization checking. You should not issue a SETROPTS RACLIST for the same class.

When an application RACLISTs a class using RACROUTE REQUEST=LIST,GLOBAL=YES, the RACLISTed profiles are stored in a data space. The data space can be shared by many applications. Applications that issue a subsequent RACROUTE REQUEST=LIST,GLOBAL=YES for the same class simply access the data space built by the first application. When all applications have relinquished their access to the data space by issuing a RACROUTE REQUEST=LIST,ENVIR=DELETE request, the data space can be deleted by issuing a SETROPTS NORACLIST(*classname*) command. The SETROPTS NORACLIST command processes not only the class specified by *classname*, but also all valid classes that share the same POSIT value as *classname*. If you issue a SETROPTS RACLIST for that class, RACF rebuilds the data space from the RACF database profiles and replaces the existing data space.

Refreshing SETROPTS RACLIST processing

SETROPTS RACLIST(*classname*) REFRESH deletes the existing RACLIST data space, and loads the discrete and generic profiles from the RACF database into a new data space. If the RACGLIST class is active and contains a profile named *classname*, the contents of the data space are written to the database as *classname*_nnnnn profiles, rebuilding them if they already existed, creating them if not.

If SETROPTS RACLIST has been issued for a general resource class and you change any profiles in the class, issue SETROPTS RACLIST(*classname*) REFRESH when you want the changes to take effect. SETROPTS RACLIST(*classname*) REFRESH can also be used to refresh classes RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES. The scope of a RACLIST REFRESH command is the class named on the command plus any other classes sharing the same POSIT value. Refer to *z/OS Security Server RACF Security Administrator's Guide* for further information.

The following example shows how to refresh SETROPTS RACLIST processing for the DASDVOL and TERMINAL classes.

```
SETROPTS  RACLIST(DASDVOL  TERMINAL)  REFRESH
```

Shared system considerations: If your installation has two or more systems sharing a RACF database, you must issue the SETROPTS RACLIST REFRESH command on all systems to have the results effective on all systems, unless RACF is enabled for sysplex communication. However, if you do not perform a refresh (by issuing the SETROPTS command with the REFRESH option) on a system sharing a RACF database and that system needs to re-IPL, a fresh copy of the RACLISTed

profiles is read from the database at IPL time. When RACGLIST profiles exist for the class, SETROPTS RACLIST(*classname*) REFRESH *must* be used for the changed profiles to become effective, even if the system is IPLed.

When RACF is enabled for sysplex communication, it propagates the SETROPTS RACLIST REFRESH command to each of the systems in the data sharing group.

GENLIST processing

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). Using GENLIST saves real storage because generic profiles are not duplicated in each user's address space. I/O is required only once to bring them into storage for all address spaces to use, instead of each address space needing to perform the I/O.

RACF uses these profile copies to check the authorization of any user who wants to access a resource the profiles protect, if RACF does not find a discrete profile for the resource in the RACF database.

To activate GENLIST processing, a user with the SPECIAL attribute issues the SETROPTS command:

```
SETROPTS GENLIST(classname...) CLASSACT(classname...)
```

Use SETROPTS GENLIST when the class contains a small number of frequently referenced generic profiles.

If you issue a SETROPTS GENLIST on one system, that action *is* propagated to other systems that share the RACF database. You do not need to issue the SETROPTS GENLIST command separately for each system.

In-storage profiles for the following classes supplied by IBM can be shared by using SETROPTS GENLIST:

APPL	FACILITY	KEYSMSTR	TERMINAL	VMMDISK
CPSMOBJ	FIELD	LOGSTRM	TMEADMIN	VMNODE
DASDVOL	INFOMAN	RRSFDATA	VMBATCH	VMRDR
DCEUUIDS	JESJOBS	SDSF	VMCMD	VMSEGMT
DSNR				

Generic profiles for the DATASET class will continue to be created within each address space and chained off the ACEE.

Refreshing in-storage generic profiles

You might want to use GENERIC REFRESH after changing a generic profile that protects a specific data set. However, extensive use of GENERIC REFRESH can adversely affect system performance.

You can refresh in-storage generic profiles by specifying both the GENERIC and REFRESH operands on the SETROPTS command. When you specify both GENERIC and REFRESH, you also specify one or more classes for which you want RACF to refresh in-storage generic profiles. This causes all the in-storage generic profiles within the specified general resource class (except those in the global access checking table) to be replaced with new copies from the RACF database. The following example shows how to refresh in-storage generic profiles for the DATASET and TERMINAL classes.

```
SETROPTS  GENERIC(DATASET  TERMINAL)  REFRESH
```

You must issue this command each time you want RACF to perform the refresh process.

If you specify GENERIC(*), RACF refreshes profile lists for the DATASET class and all active classes in the class descriptor table except group resource classes (such as GTERMINL and GDASDVOL).

SETROPTS REFRESH processing on shared systems: The refresh operation for SETROPTS processing applies only to the system on which you issue the SETROPTS command, unless RACF is enabled for sysplex communication. If RACF is not enabled for sysplex communication and two or more systems in your installation are sharing a RACF database, you must issue the SETROPTS command on each system to have the refresh done on all systems. When RACF is enabled for sysplex communication, RACF propagates the command to each of the systems in the data sharing group. However, if you do not perform a refresh (issue the SETROPTS command with the REFRESH option) on a system sharing a RACF database and that system needs to re-IPL, the refresh takes effect on that system when re-IPL is performed.

Using SETROPTS INITSTATS and SETROPTS STATISTICS

An installation can record two types of RACF statistics. One is INITSTATS, which records user logon information; the other is STATISTICS, which records access to resources in specific classes that are protected by discrete profiles. There are several initial recommendations:

- When a new RACF database is initialized, the default is INITSTATS on.

  It is recommended that you use INITSTATS because it allows you to use other options to provide additional security at logon.
- When a new RACF database is initialized, the default is STATISTICS off for all classes.

  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 the SETROPTS command in *z/OS Security Server RACF Command Language Reference*.

Note: If you are sharing a database and are using in-storage data blocks, statistical information might not be accurate.

INITSTATS processing

INITSTATS records statistics on all user profiles in the system. INITSTATS also allows your installation to take advantage of the SETROPTS INACTIVE option and the REVOKE, HISTORY, and WARNING options of SETROPTS PASSWORD. Only users with SPECIAL authority can control the recording of INITSTATS.

INITSTATS records the following:

- The date and time RACF processes a RACROUTE REQUEST=VERIFY (for example, logon or batch job initiation) for a particular user.
- The number of RACROUTE REQUEST=VERIFYs for a user to a particular group.
- The date and time of the last RACROUTE REQUEST=VERIFY for a user to a particular group.

These statistics are recorded on both the primary and backup databases the first time each day that the user uses the system, each time the user changes his or her password, and each time the user enters the correct password after having previously entered an incorrect password. The recording occurs whenever the backup is active, even if ICHRDSNT specifies that statistics should not be maintained on the backup database. This ensures that users are not accidentally revoked if you need to switch to the backup database.

At all other times these statistics are recorded only to the primary database.

Recommendations on using INITSTATS: Although INITSTATS affects performance because of I/O to the database, it is recommended that INITSTATS stay on. You can then use the SETROPTS operands INACTIVE and PASSWORD. For additional information, see *z/OS Security Server RACF Security Administrator's Guide*.

When RACF is enabled for sysplex communication, the performance of recording the statistics to the backup database should improve because RACF has in-storage buffers for the backup database.

STATISTICS processing

The STATISTICS option permits an installation to record statistics on discrete profiles to see how their respective data sets and resources within specific resource classes are being used. Statistics are not recorded for profiles that are loaded into storage by RACROUTE REQUEST=LIST or SETROPTS RACLIST. Only a user with SPECIAL authority can control the recording of STATISTICS.

STATISTICS does the following:

- 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. It can be difficult to compare the two sets of statistics meaningfully, unless you understand how RACF maintains the statistics. See *z/OS Security Server RACF Security Administrator's Guide* for more information.
- If a specific resource has unique security concerns, you should protect it with a discrete profile.

  To see how that resource is being accessed and how many times it is being accessed, you can initiate STATISTICS. 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 within the various resource classes, turning on STATISTICS might negatively affect performance.

Recommendations on using STATISTICS: Do not use a discrete profile and the STATISTICS option to protect a heavily accessed resource. Doing so increases I/O to the database and decreases system performance, because STATISTICS are kept on all discrete profiles in the same resource class.

If you wish to keep statistics for some data sets, protect those with discrete profiles, and use generic profiles to protect the remainder of your data sets.

There is a relationship between STATISTICS and the POSIT number in the class descriptor table (CDT). Several classes in the CDT might share the same POSIT number because the resource classes have similar processing needs. Because those classes share the same POSIT number, if you activate STATISTICS on the discrete profiles in one class, you simultaneously activate STATISTICS on all discrete profiles in the classes that share the same POSIT number.

We recommend that you not record statistics on your backup database, because your system performance might decrease sharply. However, if it is critical that you record statistics on your backup database, consider enabling RACF for sysplex communication. This causes RACF to allocate in-storage buffers for the backup database, which should improve the performance of recording the statistics. Also, make sure that your backup database is an exact copy of the primary (made by using IRRUT200), as this further improves the performance of recording the statistics.

See the SETROPTS command in *z/OS Security Server RACF Command Language Reference* for further information.

Identification, verification, and authorization of user IDs

RACF processing determines whether work is allowed to enter the system and who is authorized to access resources in the system.

The following RACROUTE requests are used repeatedly for these tasks:
- RACROUTE REQUEST=VERIFY or VERIFYX
- RACROUTE REQUEST=SIGNON
- RACROUTE REQUEST=AUTH

User identification and verification

RACROUTE REQUEST=VERIFY or VERIFYX processing

The RACROUTE REQUEST=VERIFY function does identification and verification of users and determines whether work is allowed to enter the system. Some of the events that can cause VERIFY request processing to occur are:
- Logons to TSO, IMS™, or CICS
- Submitting a batch job
- Sending data sets to the printer (if WRITER class is active)
- Processing certain operator commands (if OPERCMDS class is active)
- Running APPC/MVS transactions

Some of the checks done by REQUEST=VERIFY processing are:
- Surrogate checking
- Terminal-authorization and port-of-entry checking
- JESJOBS checking

For certain callers (RACROUTE REQUEST=VERIFY), specifying SYSTEM=YES on the RACROUTE requests can provide better performance. For more information, see *z/OS Security Server RACROUTE Macro Reference*.

Specifying a session type of OMVSSRV can also improve performance. When a RACROUTE REQUEST=VERIFY or RACROUTE REQUEST=VERIFYX macro specifies the OMVSSRV session type:
- RACF updates the date and time of last user access at most once a day.
- RACF creates audit records for RACROUTE REQUEST=VERIFY,ENVIR=CREATE only when the macro specifies a new or incorrect password, or the user ID is revoked.

For more information, see *z/OS Security Server RACROUTE Macro Reference*.

RACROUTE REQUEST=SIGNON processing

The RACROUTE REQUEST=SIGNON function is provided to build a list of identified and verified users in an LU 6.2 persistent verification environment.

Programs that query the list entries can obtain a verified user's security environment. For more information, see *z/OS Security Server RACROUTE Macro Reference*.

Improving verification performance using VLF

RACF processing of user verification requests might be improved by using virtual lookaside facility (VLF). During verification, RACF can save ACEEs by using VLF, and retrieve them on subsequent verification calls for the same user.

Performance improves through pathlength reduction and elimination of I/O to the RACF database. If multiple requests from repetitive tasks are made (for example, when batch jobs are going through the system), there is likely to be a match in VLF for the ACEE being built.

See also "Removing information from VLF" on page 68.

For RACF to begin saving and retrieving ACEEs, you must define the IRRACEE class to VLF. For more information, see "ACEEs and VLF considerations" on page 67. For information on VLF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

RACROUTE REQUEST=AUTH processing

Whenever a user attempts to access a resource, the system calls RACF to perform authorization checking. During normal RACROUTE REQUEST=AUTH processing, RACF always authorizes full access to a user's own data (based on the high-level qualifier) and references the corresponding profile to see whether statistics or logging is indicated.

An installation can bypass normal REQUEST=AUTH processing by using the global access-checking facility. When global access checking allows a request, RACF performs no I/O to the RACF database, performs no logging, and maintains no statistics. As a result, global access checking provides you with a fast way to allow access to selected resources.

A global access table for the DATASET class is recommended because of the frequency of AUTH requests that can occur.

- If your installation is using enhanced generic naming (EGN) support, you can enter `&RACUID.**/ALTER` in the global access checking table.
- If your installation is *not* using EGN support, and most users access their own data sets, you should include the entry `&RACUID.*/ALTER` in the global access checking table to bypass normal processing for a user's own data sets.

In addition, if generic profile checking is active during authorization checking, RACF builds lists of generic profiles in storage to be referenced repeatedly by the RACROUTE REQUEST=AUTH function. The use of generic profiles can reduce the size of the RACF database, reduce the time and effort needed to maintain profiles, and minimize the frequency of I/O requests to the RACF database.

However, these benefits are lost if too many generic profiles are defined:

- Within a general resource class
- With the same high-level qualifier in the DATASET class

RACF generic profiles work best when you have multiple resources protected by a single profile.

Note that RACF authorization checking bypasses data-set password checking. RACF also eliminates the need for an operator message requesting a password, for password-protected DASD data sets.

RACROUTE REQUEST=FASTAUTH processing

RACROUTE REQUEST=FASTAUTH uses the resident profiles to perform authorization checking. RACROUTE REQUEST=FASTAUTH gathers no statistics, issues no service calls (SVCs), and is branch-entered by the resource manager. The RACROUTE REQUEST=FASTAUTH service is SRB-compatible.

If you use RACROUTE REQUEST=FASTAUTH rather than RACROUTE REQUEST=AUTH, you can improve your application's performance. FASTAUTH is particularly useful for applications that have stringent performance requirements. But FASTAUTH processing does complicate your application coding: if your application does not run in system key or supervisor state, you must use RACROUTE REQUEST=LIST to load the profiles into storage. If you use RACROUTE REQUEST=LIST,ENVIR=CREATE,GLOBAL=NO to load the profiles into storage, then you must issue REQUEST=LIST,ENVIR=DELETE to delete the profiles when you are done. In addition, if your application is long-running, you might need to supply a "refresh" mechanism in case the security administrator has changed a profile.

However, if you issue RACROUTE REQUEST=LIST,GLOBAL=YES to load the profiles into a data space, the administrator can refresh the profiles with SETROPTS RACLIST REFRESH. Therefore, your application does not have to provide a method to refresh the profiles. Likewise, REQUEST=LIST,ENVIR=DELETE deletes the application's access to the profiles, but not the data space; SETROPTS NORACLIST deletes the data space. If RACGLIST profiles are being maintained in conjunction with RACLISTed data, a SETROPTS NORACLIST deletes the RACGLIST *classname*_nnnnn profiles on the database.

If your application runs in an authorized state, it can use profiles brought into storage by SETROPTS RACLIST for authorization checking, and no RACROUTE REQUEST=LIST is required.

Using generic profiles

In each address space, RACF keeps up to four lists of generic profiles that have been referenced. Each list comprises one DATASET high-level qualifier, or one general resource class based on the value of KEYQUAL in the CDT for that class (assuming the class is not RACLISTed in some way).

When RACF needs to reference a set of generic profiles that are not present in the address space, the oldest list is deleted and the new list replaces it. The performance impact of doing this can be especially important during the OPEN for a concatenated DD statement. If possible, group data sets with the same high-level qualifier together in the concatenation, so that RACF does not need to read the same list of generics multiple times. Also, consider using global access checking for commonly referenced data sets, because RACF does not need to use the generic profiles if the access is granted by global access checking.

When RACF loads the list of generic profile names, significant I/O to the RACF database might occur. Therefore, the number of generic profiles within a data set high-level qualifier or general resource class should be kept as small as practical,

which might suggest the use of discrete profiles instead of generics. The performance of generics in RACF is optimized for the case where each generic profile protects several (possibly many) resources for the average case.

Mapping UIDs to user IDs and GIDs to group names

The virtual lookaside facility (VLF) is used to map z/OS UNIX user identifiers (UIDs) to user IDs and z/OS UNIX group identifiers (GIDs) to group names, and should be active when running z/OS UNIX System Services.

Note: VLF can be used for identity mapping with a RACF database created before OS/390 Release 10. It is not needed when using an alias index at application identity mapping stage 3.

If VLF is not active, requests for UID-to-user ID mapping and GID-to-group name mapping default to searching the RACF database on each request. This significantly degrades performance of these functions. It could also affect other systems in a complex where more than one system is sharing the RACF database, because of the increased I/O to the database. Running without VLF active should be done only when it is necessary to stop VLF to make changes to it.

When VLF is active but a UID or GID is not found in VLF, RACF can determine the corresponding user ID or group name by accessing one profile in the UNIXMAP class. RACF adds the mapping to VLF if it finds it in the UNIXMAP class.

For RACF to begin using VLF for UID and GID mapping, you must define the IRRGMAP and IRRUMAP classes to VLF and VLF must be active. For more information, see “VLF considerations for mapping UIDs and GIDs” on page 69. For information on VLF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

For RACF to use the UNIXMAP class, the class must be active. For more information on the UNIXMAP class, see *z/OS Security Server RACF Security Administrator's Guide*.

z/OS UNIX System Services applications

To improve the performance of z/OS UNIX System Services applications that use thread level security services such as the pthread_security_np service, RACF can use the virtual lookaside facility (VLF) to cache user security packets (USPs). Use of VLF to cache USPs is optional. For RACF to use VLF to cache USPs, you must define the IRRSMAP class to VLF, and VLF must be active. For information on how to do this, see “VLF considerations for caching user security packets (USPs)” on page 69.

Large profiles

Large profiles can degrade performance in the following ways:

- When RACF brings a profile into storage, a large profile takes up a significant portion of the in-storage buffer. As a result, the rate of I/O to the RACF database increases.
- If an update to a profile causes the profile to require an additional block in the RACF database, the larger the profile's size the further RACF is likely to have to go in the database to find enough contiguous blocks for the updated profile, increasing the distance between the profile and its index block. As a result,

searches for the profile become longer and longer as you update the profile. In addition, the database becomes more fragmented, requiring that you run the IRRUT400 utility more often to reorganize it.

Large groups

Although you can connect approximately 5900 users to a group, large groups can degrade performance. Every time RACF does processing related to a group, RACF brings the group's profile into storage. Processing of the RESOWNER field in a data set profile can require that RACF read a group profile into storage. Because large groups have large group profiles, when RACF brings a profile for a large group into storage, or updates the profile, it has the same effects on performance discussed in "Large profiles" on page 34.

Universal groups

A universal group is a large group that can include more than 5900 users. The number of users connected to the universal group does not affect performance.

Chapter 3. RACF customization

Specifying RACF database options 37
The data set name table 37
Table format 38
RACF sysplex communication 39
Effects of not using a data set name table 40
Emergency data set name tables 40
Sysplex considerations 40
Selecting the number of resident data blocks 41
Data set name table examples 42
The database range table 45
Table format 46
Database range table example 47
Specifying resource-class options 48
The class descriptor table (CDT) 48
Installation-Defined Classes and the RACF Router Table 50
Adding installation-defined classes to the class descriptor table 50
Changing an installation-defined class in the class descriptor table 51
Deleting an installation-defined class from the class descriptor table . . . 52
The RACF router table 52
Adding an entry to the RACF router table 53
Password authentication options 54
The RACF DES algorithm 54
How the RACF DES algorithm works 54
Migrating to the RACF DES authentication algorithm 54
Using the DES algorithm without the two-step method of checking 55
Using the masking algorithm 56
Using your own authentication algorithm 56
PassTicket authentication 56
How RACF processes the password or PassTicket 56
Changing the RACF report writer options (ICHRSMFI module) 57
Customizing the RACF remote sharing facility 59
Customizing the RACF/DB2 external security module 59

Specifying RACF database options

You can specify options for the RACF database by using the following:

- The data set name table, which describes the data sets in the RACF database to RACF and allows you to select the number of resident data blocks, the sysplex communication option, and the default data sharing mode
- The database range table, which determines which data set in the RACF database to access for a particular profile

The data set name table

The data set name table (ICHRDSNT) is an installation-defined load module that describes the data sets in the RACF database to RACF. This table contains entries describing each data set in the RACF database and its backup data set. ICHRDSNT is also used to configure systems for the sysplex communication and data sharing options.

A data set's position in this table corresponds to the data set number in the range table. If a data set is named in the data set name table, it *must* be referenced in the

© Copyright IBM Corp. 1994, 2004

range table. If the name table does not match (that is, has more entries than) the range table, RACF does not become active during the IPL.

RACF can have as many as 90 data sets in the primary database and 90 associated data sets in the backup database.

Table format

The first byte of the name table is a binary number indicating the number of entries in the table. Each entry consists of:

- A 44-byte data set name (primary)
- A 44-byte data set name (backup)
- A 1-byte resident data-block count field
- A 1-byte flag field

A 44-byte data set name (primary): The first data set name identifies a data set in the primary database. The first primary RACF data set activated is considered to be the *master primary RACF data set.* If your installation has specified this field with an asterisk (*), RACF prompts the operator during RACF initialization, to supply the data set name.

A 44-byte data set name (backup): The second data set name identifies an associated data set in the backup database. If your installation has specified this field with an asterisk (*), RACF prompts the operator again. A blank data set name field indicates the absence of a backup data set for the associated primary data set for this IPL.

A 1-byte resident data-block count field: The resident data-block count field specifies the maximum number of index, block-availability-mask (BAM), and profile blocks to be kept resident for the primary data set while the database is active. In addition, if RACF is enabled for sysplex communication, RACF uses 20% of this number as the number of resident blocks for the backup data set. See “Selecting the number of resident data blocks” on page 41.

A 1-byte flag field: The format of the flag field is as follows:

Bit Setting	Meaning
00..	No updates are to be duplicated on the backup data set. This is the default setting if you do not provide ICHRDSNT.
10..	All updates, but no statistics, are to be duplicated on the backup data set. This is the recommended setting if you provide ICHRDSNT. If SETROPTS INITSTATS is on, a limited subset of statistics is maintained on the backup: • The first time each day that the user logs on when SETROPTS INACTIVE is in effect • The time and date of password changes during logon • The time and date a user enters a correct password after having entered an incorrect one SETROPTS INITSTATS allows RACF to continue processing revoke dates if you need to switch to your backup database.
11..	All updates, including statistics, are to be duplicated on the backup data set.
.... *xx*..	Sysplex communication and data sharing bits. These bits need to

be set in the first entry only of the data set name table. RACF ignores these bits in subsequent data set name table entries. In releases before RACF 2.1, these bits are ignored. These bits can have the following values:

Bit Setting	Meaning
00	RACF is not enabled for sysplex communication.
10	RACF is enabled for sysplex communication, and requests non–data sharing mode at IPL.
01	RACF requests data sharing mode at IPL. Because this requires sysplex communication, RACF is also enabled for sysplex communication. (This bit setting is treated the same as the 11 setting.)
11	RACF is enabled for sysplex communication, and requests data sharing mode at IPL.

When RACF is enabled for sysplex communication, it allocates resident data blocks for the backup database.

All systems in the data sharing group must be in the same mode, either all data sharing or all non–data sharing mode. As a result, the mode specified here may be overridden at IPL time to the mode in use by the other members of the data sharing group.

See “Sysplex considerations” on page 84 for more information on the sysplex communication and sysplex data sharing options.

....1 Formerly controlled the resident data-block option for the primary database. Beginning with RACF 1.9.2, the resident data-block option is always used and the setting for this bit is ignored.

This table resides in SYS1.LINKLIB or any other APF-authorized linklist library. It must be linked with RMODE(24).

The RACTABLE member of SYS1.SAMPLIB contains a sample MVS job that creates a RACF data set name table; also see the program directory shipped with z/OS.

RACF sysplex communication

RACF sysplex communication requires the use of an installation-defined data set name table.

- All RACF members of the data sharing group must use compatible data set name tables. RACF enforces use of the same data sets when RACF is enabled for sysplex communication. It is recommended that systems sharing the same RACF database ensure their use of a compatible data set name table by accessing ICHRDSNT from a common library.
- The number of resident data blocks specified for buffers can differ between systems in the data sharing group.

Note: When RACF is enabled for sysplex communication, the data set names and the statistics and backup flags are defined by the first system to be assigned to the data sharing group. Any subsequent systems joining the data sharing group use the same data set names, statistics, and backup flags as the first system. Therefore, even if you specify an asterisk (*) in the data set name

table for either the primary or the backup data set name, the operator on subsequent systems is not prompted for a name.

Effects of not using a data set name table

If at IPL RACF does not find an installation-supplied data set name table (and the master JCL does not point to a RACF database with a SYSRACF DD statement), RACF prompts not only for a primary RACF database, but also for a backup database. However, this results in a primary database with only 10 data blocks, an inactive backup database, and poor system performance.

Note: MSTJCL*xx* recognizes only the primary database when the SYSRACF DD statement is provided. For this reason, we recommend using a data set name table instead of MSTJCL*xx*.

Emergency data set name tables

In an emergency (for example, if you cannot access your RACF database) you can allow the operator to specify data set names and still have the benefit of using a data set name table. One suggested method:

1. Create two load libraries, for example, SYS1.ICHRDSNT.NORMAL and SYS1.ICHRDSNT.EMRGNCY.
2. Put your production ICHRDSNT module in SYS1.ICHRDSNT.NORMAL.
3. Put an emergency ICHRDSNT module in SYS1.ICHRDSNT.EMRGNCY.

   It should be identical with your production ICHRDSNT module except that it should have an asterisk (*) for each data set name.
4. Create two LNKLST*xx* members in SYS1.PARMLIB, for example, 00 (normal) and EM (emergency).
5. Put SYS1.ICHRDSNT.NORMAL at the end of 00, and SYS1.ICHRDSNT.EMRGNCY at the end of EM.

With this procedure, for a normal IPL, the operator is not prompted. However, in an emergency, specifying LNKLST=EM during IPL allows the operator to enter the data set names to be used.

Sysplex considerations

If you use the above procedure when your current RACF system is enabled for sysplex communication, and the emergency ICHRDSNT is also requesting sysplex communication, you must re-IPL the entire RACF data sharing group to cause all the sharing systems to use the same new set of data sets. To do this:

1. Bring down all systems in the sysplex that are using sysplex communication. If you don't bring down all the systems, the operator is not prompted and the system uses the same data set names as the rest of the data sharing group.
2. When all the systems are down, re-IPL them. If an asterisk (*) has been specified for the data set names, one system prompts you for the data set names. The other systems use the same data set names.

We recommend that you also have an emergency ICHRDSNT that has the sysplex communication and data sharing bits OFF (either instead of the above emergency ICHRDSNT with either bit on, or in addition to it). This will allow you to bring up a single necessary member in non-sysplex-communication/datasharing mode, which might be necessary in some recovery scenarios.

Attention: This ICHRDSNT should specify a database other than the one used in your sysplex. (Perhaps a weekly backup of your sysplex database could be specified.) This is necessary because serialization used in non sysplex-communication/datasharing environment is incompatible with serialization in a sysplex-communication/datasharing environment. Bringing up a non sysplex-communication/datasharing system against your main sysplex database is likely to result in database corruption.

Selecting the number of resident data blocks

To avoid database I/O, RACF buffers database blocks in resident storage. In the data set name table (ICHRDSNT), you can specify the number of resident data blocks for each primary RACF data set. This keeps any type of data block (profile and BAM blocks as well as index blocks) resident. An installation can specify from 0 to 255 resident data blocks. If RACF is not enabled for sysplex communication, the default value is 10 resident data blocks if you do not provide ICHRDSNT. If enabled for sysplex communication, RACF enforces a minimum of 50 resident data blocks for a primary data set and 10 (20% of 50) for a backup data set. For best performance, specify as large a number of buffers as you can afford, preferably 255.

Resident data blocks reduce the amount of I/O that is required to service the RACF database. While the number of blocks remains the same for the duration of an IPL, the function is dynamic because, at any time, the most frequently used blocks are kept in storage.

The number of bytes (per primary data set) of ECSA storage used by the data blocks is 3248 + (4144 x the number of blocks). If RACF is enabled for sysplex communication, an additional 3248 + (4144 $\times$ .2 $\times$ the number of blocks) is used for the backup data blocks. During IPL, RACF obtains the storage for the number of buffers specified in the data set name table. The RACF manager then keeps track of when each buffer was used last. The RACF manager does different processing for shared and non-shared databases.

Shared RACF database:

- The change count in the inventory control block (ICB), corresponding to the block type (profile or index), is updated whenever a block is updated.
- In data sharing mode, RACF uses MVS XES services to communicate the changes needed to blocks in the local buffers. This method provides a more granular scheme for invalidating buffers.
- If RACF is not running in data sharing mode, index block buffers are marked as out-of-date if the change count in the ICB for that level differs from the change count in the in-storage buffer.
- If RACF is not running in data sharing mode, profile buffers are marked as out-of-date if the change count in the ICB for profile blocks differs from the change count in the in-storage buffer.
- If you are sharing a database and using in-storage data blocks, statistical information in RACF profiles might not be accurate.

Note: If the RACF database is shared, you do not need to specify the same number of resident data blocks for all systems that share the RACF database. However, if you are using sysplex data sharing, you must define coupling facility structure definitions large enough for the largest specification made for that database by any of the sharing systems. See “Defining RACF structures for the coupling facility” on page 90 for information on coupling facility structure definitions.

Non-shared RACF database:

- When reading a block, the RACF manager first searches the in-storage buffers for a valid copy of the block. If it finds one, it uses it. If it doesn't find a valid copy, the RACF manager obtains an in-storage buffer from the pool of buffers, reads the data block into that buffer, and retains the data block in storage after the I/O operation.
- When updating a block, the RACF manager searches the in-storage buffers for a copy of the block. If it doesn't find one, it obtains an in-storage buffer from the pool of buffers.

  When a block is updated, RACF always performs an I/O operation so that the RACF database has an up-to-date version of that block.

When getting a buffer from the pool, the RACF manager attempts to get a buffer that is empty or contains an out-of-date block. (A block is only out-of-date in a shared database system.) If it finds none, the manager takes the buffer containing the least-recently used block.

Data set name table examples

Example 1—using a split database: Assume that your database has been split into three parts and that your installation arranges its database profiles in the three data sets as follows:

- RACF.RACFDS1—test data sets or resource profiles
- RACF.RACFDS2—production data sets or resource profiles
- RACF.RACFDS3—system data sets or resource profiles

For recovery, the installation wants a backup data set for each primary RACF data set. However, the backup of the data sets is different:

- For RACF.RACFDS1, all updates to the primary data set, except statistics, are duplicated in the backup data set.
- For RACF.RACFDS2 and RACF.RACFDS3, all updates to the primary data set, including statistics, are duplicated in the backup data set.

The following data set name table correctly follows these criteria:

AL1(3)
Number of primary RACF data sets

CL44'RACF.RACFDS1'
Name of first RACF data set in the primary database (test data sets)

CL44'RACF.BACKUP1'
Name of first RACF data set in the backup database

AL1(20)
Number of resident data blocks

XL1'80'
Flags; all updates other than statistics updates are to be duplicated in the backup data set

CL44'RACF.RACFDS2'
Name of second RACF data set in the primary database (production data sets)

CL44'RACF.BACKUP2'
Name of second RACF data set in the backup database

AL1(10)
Number of resident data blocks

XL1'C0'
Flags; all updates, including statistics, are to be duplicated in the backup data set

CL44'RACF.RACFDS3'
Name of third RACF data set in the primary database (system data sets)

CL44'RACF.BACKUP3'
Name of third RACF data set in the backup database

AL1(255)
Number of resident blocks

XL1'C0'
Flags; all updates, including statistics, are to be duplicated in the backup data set

Figure 2. ICHRDSNT example 1—for three data sets

Example 2—Using RACF sysplex data sharing: Your installation has allocated two primary RACF data sets:
- SYS1.RACFP1—1st primary
- SYS1.RACFP2—2nd primary

For recovery, the installation also wants a backup data set for each primary RACF data set, although the backup of the data sets is different:
- For SYS1.RACFP1, all updates to the primary database, except statistics, are duplicated on the backup database.
- For SYS1.RACFP2, all updates to the primary database, including statistics, are duplicated on the backup database.

Notes:
1. The sysplex communication bit and the RACF data sharing mode bit are specified only in the entry for the first data set.

2. RACF allocates 51 (255 × .20) resident data blocks for the first backup data set buffer and 10 (50 × .20) for the second.

The following data set name table correctly follows these criteria:

AL1(2)
Number of data sets in the primary RACF database

CL44'SYS1.RACFP1'
Name of first primary data set

CL44'SYS1.RACFB1'
Name of first backup data set

AL1(255)
Number of resident data blocks (for primary database)

X'8C' Flags; all updates other than statistics are to be duplicated in the backup data set. The sysplex communication and data sharing bits are both turned on.

CL44'SYS1.RACFP2'
Name of second data set in the primary RACF database

CL44'SYS1.RACFB2'
Name of second data set in the backup RACF database

AL1(50)
Number of resident data blocks

X'C0' Flags; all updates, including statistics, are to be duplicated in the backup data set.

Figure 3. ICHRDSNT example 2—data sharing option and split database

Example 3—Using RACF sysplex communication: Your installation has allocated two data sets for the primary RACF database:
- SYS1.RACFP1—1st primary data set
- SYS1.RACFP2—2nd primary data set

For recovery, the installation also wants a backup data set for each primary RACF data set, although the backup of the data sets is different:
- For SYS1.RACFP1, all updates to the primary data set, except statistics, are duplicated on the backup data set.
- For SYS1.RACFP2, all updates to the primary data set, including statistics, are duplicated on the backup data set.

Notes:
1. The sysplex communication bit is specified only in the entry for the first data set. The data sharing bit is off.
2. RACF allocates 51 (255 × .20) resident data blocks for the first backup data set buffer and 10 (50 × .20) for the second.

The following data set name table correctly follows these criteria:

AL1(2)
Number of data sets in the primary RACF database

CL44'SYS1.RACFP1'
Name of first data set in the primary RACF database

CL44'SYS1.RACFB1'
Name of first data set in the backup RACF database

AL1(255)
Number of resident data blocks (for primary data set)

X'88' Flags; all updates other than statistics are to be duplicated on the backup data set. The sysplex communication bit is on, and the data sharing bit is off.

CL44'SYS1.RACFP2'
Name of second data set in the primary RACF database

CL44'SYS1.RACFB2'
Name of second data set in the backup RACF database

AL1(50)
Number of resident data blocks

X'C0' Flags; all updates, including statistics, are to be duplicated in the backup data set.

Figure 4. ICHRDSNT example 3—sysplex communication and split database

The database range table

The range table (ICHRRNG) is a load module. This table determines in which data set of the RACF database RACF places each profile. This table must reside in a link-pack area (LPA) library or in a modifiable link-pack area (MLPA) library, such as SYS1.LPALIB. It must be linked with RMODE=24.

All systems sharing a database must use the same database range table. One way to ensure that each system uses the same table is to put the table in a common link library.

If RACF is enabled for sysplex communication, it verifies at IPL that a local system's range table matches the range table used by the rest of the data sharing group. If there is a mismatch, it uses the table used by the group.

RACF provides a default range table with the following values:

F'1' Number of range values

XL44'00' Range start value

AL1(1) Data set number

This table assumes the RACF database has one data set containing all profiles. If you wish to split your database, you must replace the RACF load module with your own. Do this by creating a source file, assembling the file, and link-editing it. You should use an SMP/E USERMOD to do the assembly and link-edit.

Table format

The first fullword of the range table is a binary number indicating the number of entries in the table. Each entry consists of:

XL44'00' Range start value

AL1(*n*) where *n* is the data set number

The first **range start value** must contain 44 bytes of binary zeros. You must arrange the table in ascending order of the 44-byte strings.

The one byte **data set number** indicates the data set's relative position in the data set name table (ICHRDSNT).

If zero is specified for the data set number it indicates that the range is not associated with a data set. The RACF manager returns a code of 28 when an attempt is made to access an entity in such a range.

If the data set number is nonzero, RACF assigns the profile for each entry name that falls within the range represented by the 44-byte string to the data set in the RACF database with the corresponding number in the ICHRDSNT table.

When constructing a range table, you must consider the way in which RACF constructs the internal form of the names of certain types of entries. The RACF manager uses only the internal forms of these entry names; therefore, you must also use the internal names when you construct the range table.

Internal profile naming for general resource classes: The form of the entry name the RACF manager uses for general resource classes consists of prefixing 9 characters to the beginning of the entry name. It uses the 8 characters of the class name (padded with trailing blanks if the class name is shorter than 8 characters), plus a dash. For example, a TAPEVOL named DATA1 becomes TAPEVOL -DATA1, and a DASDVOL named DATA1 becomes DASDVOL -DATA1.

RACF modifies generic profile names internally, as follows:

- For DATASET profiles, the first delimiter (a period) is converted to X'01'. In addition, RACF modifies the generic characters.
- For general-resource classes, the hyphen (-) that is added internally is converted to X'02'. In addition, RACF modifies the generic characters.

Note: You must be careful with the TAPEVOL class. Although there is no requirement that all entries in this class be directed to the same data set in the RACF database, you must direct all the *members* of any tape volume set to a single data set in the RACF database. The RACF manager returns a code of 60 if you attempt to add to a tape volume set a volume whose profile is not assigned by the range table to the same data set in the RACF database.

Internal profile naming for alias index entries: The form of the entry name the RACF manager uses for alias index entries consists of prefixing 3 bytes of non-EBCDIC values to the beginning of the entry name. These 3 bytes define the characteristics of the alias index entry. Specifically, the bytes represent the template number, segment number, and field number associated with the entry. The following table illustrates the hexadecmial values in effect for a particular alias field:

Table 1. Alias index entry values

Alias index field specified	Alias index internal entry name
Group defined with an OMVS GID value	X'010302' followed by GID value
User defined with an OMVS UID value	X'020802' followed by UID value
User defined with an LNOTES SNAME	X'020C02' followed by SNAME
User defined with an NDS UNAME	X'020D02' followed by UNAME

For more details concerning the first 3 bytes of an alias index entry and how to translate the values into specific characteristics, see the entry name description in *z/OS Security Server RACF Diagnosis Guide*.

Database range table example

An installation wants all the alias index entries for UIDs and GIDs to reside on data set 2, all profile names beginning with "GRPA" through "GRPF" and "TEST1" through "TEST8" to reside on data set 2, all profile names beginning with "SYS1" to reside on data set 3, and all the remaining data to reside on data set 1. The following range table meets these criteria:

```
   DC F'9'        Number of range values
A  DC XL44'00'    Range Start
   DC AL1(1)      Data set number
B  DC XL44'00'
   ORG B
   DC XL3'010302' Range start for GID alias indexes
   ORG
   DC AL1(2)
C  DC XL44'00'
   ORG C
   DC XL3'020C02' Range start for LNOTES SNAME alias indexes
   ORG
   DC AL1(1)
D  DC XL44'00'
   ORG D
   DC C'GRPA'     Range start GRPA
   ORG
   DC AL1(2)      Data set number
E  DC XL44'00'
   ORG E
   DC C'GRPG'     Range start GRPG
   ORG
   DC AL1(1)      Data set number
F  DC XL44'00'
   ORG F
   DC C'SYS1'     Range start SYS1
   ORG
   DC AL1(3)      Data set number
G  DC XL44'00'
   ORG G
   DC C'SYS2'     Range start SYS2
   ORG
   DC AL1(1)      Data set number
H  DC XL44'00'
   ORG H
   DC C'TEST1'    Range start TEST1
   ORG
   DC AL1(2)      Data set number
I  DC XL44'00'
   ORG I
   DC C'TEST9'    Range start TEST9
   ORG
   DC AL1(1)      Data set number
```

Figure 5. ICHRRNG example for three data sets

Specifying resource-class options

The resources that RACF can protect are data sets, users, groups and general resources. Classes of general resources are defined in the class descriptor table (CDT). For each general resource class, there is a unique entry in the table.

The class descriptor table (CDT)

The class descriptor table (CDT) contains information that directs the processing of general resources. RACF references the class descriptor table whenever it receives a resource class name other than DATASET, USER or GROUP.

All resource classes are represented in the two load modules of the class descriptor table.

- ICHRRCDX is the name for the class entries supplied by IBM.

- ICHRRCDE is the name for the installation-defined class entries.

Each class supplied by IBM is a CSECT in load module ICHRRCDX.

Note: Do not delete or modify any of the class entries in ICHRRCDX. For a list of the classes that IBM supplies, see Appendix A, "Supplied CDT classes," on page 339.

RACF processing references the class descriptor table whenever a class name is received as input. Each class, if defined on multiple systems, must be defined identically on all systems using the class. If the classes are defined differently, unpredictable results can occur when you change the SETROPTS options for the class.

Installations sharing a database do not need identical class descriptor tables, but they must be compatible. If the same class is present on multiple systems, it must have the same attributes; for example, the POSIT numbers must be the same. Therefore, if systems X and Y are sharing a database, and system X has a class descriptor table with classes a, b, and c, and system Y has a class descriptor table with classes a, b, c, d, e, and f, the classes a, b, and c must be defined identically on both systems. However, system Y may have classes d, e, and f that are not defined on system X. Note that when RACF is enabled for sysplex communication, to allow flexibility when adding new classes to the class descriptor table, RACF does not enforce consistency in the class descriptor table as it does with the data set name table and the range table.

If you have different releases of RACF, you can share a database without adding the new classes for the higher-level system to the class descriptor table on the lower-level system. For example, if you are sharing a database between the RACF component of the z/OS Security Server and RACF 2.2, you do not have to add the new Security Server classes to the class descriptor table on the 2.2 system.

If you are using sysplex communication, RACF propagates commands. If you choose to use class descriptor tables that are compatible but not identical, command propagation might be affected. If you issue a command against a class that is not defined on the issuing system, RACF does not propagate the command. On the other hand, if you issue a command against a class that is defined on the issuing system, but RACF propagates it and finds that the class is not defined on the peer system, the command does not run on the peer system.

Systems can share the same copy of the ICHRRCDE source, if it does not contain POSIT numbers greater than 127, or if all the sharing systems are running RACF 2.2 or the RACF component of either the OS/390 SecureWay® Security Server or the z/OS Security Server. These systems can share a copy of the ICHRRCDE load module with a system running a pre-2.2 release of RACF, provided that ICHRRCDE was assembled on the lower-level system or was assembled using a pre-2.2 level of the ICHERCDE macro. Systems running RACF 2.1 or lower cannot use or share an ICHRRCDE load module after it has been assembled on a 2.2, OS/390, or z/OS system.

An installation can add, modify, or delete installation-defined class entries. The ICHERCDE macro cross-checks entries in the class descriptor table for errors. Each installation-defined class entry becomes a CSECT in load module ICHRRCDE. The ICHERCDE macro produces a CSECT for each invocation.

- If there is a CLASS operand, the CSECT name is that of the class being defined.

- If there is no CLASS operand, the CSECT name is ICHRRCDE, indicating the end of the descriptor table.

ICHRRCDE must be linked with RMODE=24.

Installation-defined names must be unique. They must not be identical to any names that IBM supplies or to other installation-defined names. The ICHERCDE macro and RACF initialization check for uniqueness.

The maximum number of entries you can have in the class descriptor table is 1024. Of these classes, 256 are reserved for IBM's use, leaving 768 for your installation's use. Classes requiring better performance should be placed towards the beginning of the table.

For information on coding the ICHERCDE macro, see the description of the ICHERCDE macro in *z/OS Security Server RACF Macros and Interfaces*.

Installation-Defined Classes and the RACF Router Table

If a class entry is added to the installation's class descriptor table, a corresponding entry should be added to the RACF router table (using the ICHRFRTB macro). See "The RACF router table" on page 52 for more information.

Adding installation-defined classes to the class descriptor table

The ICHERCDE macro generates class entries for the RACF class descriptor table. Each entry in the installation-defined class descriptor table becomes a CSECT in load module ICHRRCDE. The module resides in SYS1.LINKLIB or any other APF-authorized linklist library.

To add a class entry, specify the ICHERCDE macro for each class you are adding. Follow this procedure:

1. Produce assembler source statements to invoke ICHERCDE for each class that you are adding. For information on coding the ICHERCDE macro, see the description of the ICHERCDE macro in *z/OS Security Server RACF Macros and Interfaces*.
2. Ensure that the last entry of ICHERCDE is blank. It cannot have a CLASS operand.
3. Assemble your source.

   If the same copy of the ICHRRCDE module is being used on systems running RACF 2.1 or lower, you must either:

   - Use a RACF 2.1 or lower level of the ICHERCDE macro
   - Use the z/OS or OS/390 Security Server or RACF 2.2 level of the ICHERCDE macro and only use the resulting ICHRRCDE module on z/OS, OS/390, or RACF 2.2 systems

   If you are using POSIT numbers greater than 127, you *must* use the z/OS, OS/390, or RACF 2.2 level of ICHERCDE, and you can no longer share the resulting ICHRRCDE load module with lower-level systems.
4. Use the link-edit utility to link-edit the resulting object module into the ICHRRCDE load module, using ORDER statements for each CSECT. ICHRRCDE must be linked with RMODE=24.

   Be sure that your linkage editor ORDER statements specify ICHRRCDE as the last CSECT. Any class that does not have an ORDER statement, or any class that appears after ICHRRCDE in the output load module, is not usable.

If you install the class descriptor table with an SMP/E SYSMOD, consider assigning it a user-defined FMID, not the RACF FMID, to prevent SMP/E from deleting it during future RACF product installations.

5. Re-IPL your system for the change to take effect. In a sysplex you must re-IPL each system on which you intend to use the class before you activate the class.

If you are adding new classes to a load module previously created, you do not have to reassemble your unchanged class entries. You can use the `LKED INCLUDE SYSLMOD` statement to copy the previous version. For example, if your ICHRRCDE load module contains four classes and you are adding a fifth, here are some sample linkage editor statements to add the fifth entry to your load module:

```
//SYSLMOD   DD  DSN=SYS1.RACF.MYLOAD,DISP=OLD
//SYSOBJ    DD  DSN=SYS1.RACF.MYOBJ,DISP=OLD
//SYSIN     DD  *
      INCLUDE SYSOBJ(CLASS5)
      INCLUDE SYSLMOD(ICHRRCDE)
      ORDER CLASS1
      ORDER CLASS5
      ORDER CLASS2
      ORDER CLASS3
      ORDER CLASS4
      ORDER ICHRRCDE
      NAME  ICHRRCDE(R)
```

The RACTABLE member of SYS1.SAMPLIB contains a sample job.

For information on the class descriptor table and class entries used by CICS, see *CICS RACF Security Guide*.

Changing an installation-defined class in the class descriptor table

Changing certain fields of a class entry requires extra attention. If you are changing the POSIT value, do the following before making the change:

1. Use SETROPTS LIST and record each active option for the class.
2. Examine your classes to see if any other class is using the current POSIT value.
   - If not, use SETROPTS to turn off all the options associated with the class. This is done to reset the options associated with the POSIT value, so that you won't get any extraneous options if you later add a class using that POSIT value.
   - Otherwise, proceed to the next step.
3. After you make the change and have re-IPLed all the systems that will be using the new class, use SETROPTS to set any of the options that are still relevant for the class, using the output of the previous SETROPTS LIST as reference.

Because several classes can share the same POSIT value, changing the POSIT value might deactivate classes previously active and vice versa. (See *z/OS Security Server RACF Macros and Interfaces* for a description of POSIT numbers.)

A user who has CLAUTH authority to a class also has CLAUTH authority to all other classes with the same POSIT value. Therefore, changing the POSIT value of a class might change the set of classes to which a user has CLAUTH authority.

To modify the table, you must specify the macro for each class entry you are changing.

Follow this procedure:

1. Modify the assembler source statements that invoke ICHERCDE for each class entry.
2. Ensure that the last entry of ICHERCDE is blank. It cannot have a CLASS operand.
3. Assemble the modified source.

   If the same copy of the ICHRRCDE module is being used on systems running RACF 2.1 or lower, you must either:

   - Use a RACF 2.1 or lower level of the ICHERCDE macro
   - Use the z/OS, OS/390, or RACF 2.2 level of the ICHERCDE macro and only use the resulting ICHRRCDE module on z/OS, OS/390, or RACF 2.2 systems

   If you are using POSIT numbers greater than 127, then you *must* use the z/OS, OS/390, or RACF 2.2 level of ICHERCDE, and you can no longer share the ICHRRCDE load module with lower-level systems.
4. Use the link-edit utility to link-edit the resulting object module together with the existing ICHRRCDE load module to produce a new ICHRRCDE load module.
5. Be sure that your linkage editor ORDER statements specify ICHRRCDE as the last CSECT. Any class that does not have an ORDER statement, or any class that appears after ICHRRCDE in the output load module, is not usable.

   If you install the class descriptor table with an SMP/E SYSMOD, consider assigning it a user-defined FMID, not the RACF FMID, to prevent SMP/E from deleting it during future RACF product installations.
6. Re-IPL MVS. In a sysplex you must re-IPL each system on which you intend to use the class before you activate the class.

If you are making changes to the load module, you must reassemble the class descriptor table, or you lose the cross checking that the ICHERCDE macro performs.

Deleting an installation-defined class from the class descriptor table

You can delete a class entry from the class descriptor table by specifying the name of the class to be deleted on the link-edit REPLACE statement. For the deletion to take effect, re-IPL MVS.

You should be sure that all profiles relating to this class are deleted **before** deleting the entry from the class descriptor table.

Pay special attention to any *unique* POSIT values you use. If the class you are deleting has a *unique* POSIT value, issue a SETROPTS LIST to check what options you are using with the class—for example, CLASSACT, LOGOPTIONS, AUDIT, RACLIST, and so forth. Turn off each of the options for the class.

To illustrate: You might have activated your class. You should deactivate the class before re-IPLing your system. If you do not deactivate the class and, at a future date, you create a class with the POSIT value previously used, the class will automatically be active. The same consideration applies to each option controlled by the POSIT value.

The RACF router table

The *SAF router* is always present on a system, whether or not RACF is enabled. The resource-managing components and subsystems call the SAF router as part of

certain decision-making functions in their processing, such as access-control checking and authorization-related checking. This single SAF interface encourages the sharing of common control functions across products and across systems.

If RACF is enabled, the SAF router passes control to the *RACF router* (ICHRFR00) for certain functions. RACF uses the parameter information passed to it and the *RACF router table* to determine the appropriate RACF function to invoke.

The RACF router table contains one or more entries for each entry in the class descriptor table that IBM supplies. It also contains entries for DATASET and USER.

If you add an entry to the class descriptor table, you should add a corresponding entry to the RACF router table.

The router table consists of two modules:

- ICHRFR0X is the name of the router table module that IBM supplies.
- ICHRFR01 is the name of the installation-supplied router table module.

Note: Do not delete or modify any entries in ICHRFR0X. Doing so can produce unpredictable results.

The entries in ICHRFR01 can be locally defined resource classes and requestor/subsystem combinations.

You can have entries in the router table that do not appear in the class descriptor table.

To add an entry to the router table, use the ICHRFRTB macro. As part of its operation, the ICHRFRTB macro concatenates the values specified for the REQSTOR, SUBSYS, and CLASS operands to form a 24-character string defining the entry. For more information on the ICHRFRTB macro, see *z/OS Security Server RACF Macros and Interfaces*.

Your router table should be compatible with your class descriptor table.

Note: When RACF is enabled for sysplex communication, it does not enforce consistency of the router table as it does with the data set name table and the range table.

Adding an entry to the RACF router table

The ICHRFRTB macro generates entries for the RACF router table. The module resides in SYS1.LINKLIB (or any other APF-authorized linklist library) as ICHRFR01. It can be linked with RMODE(24) or RMODE(ANY).

To add an entry:

1. Produce assembler source statements to invoke ICHRFRTB for each entry.
2. Ensure that the last entry of ICHRFR01 has TYPE=END.
3. Assemble your source.
4. Use the linkage editor to link-edit the resulting object module into the ICHRFR01 load module. Place frequently used entries at the top.

   If you install the router table with an SMP/E SYSMOD, consider assigning it a user-defined FMID, not the RACF FMID, to prevent SMP/E from deleting it during future RACF product installations.
5. Re-IPL MVS for the change to take effect.

For information on creating ICHRFR01, see member RACTABLE in SYS1.SAMPLIB.

Password authentication options

RACF provides two algorithms for authenticating passwords: the masking algorithm and the Data Encryption Standard (DES) algorithm. The masking algorithm is the original algorithm provided with RACF. The RACF DES algorithm provides a higher level of security than the masking algorithm, and is the algorithm we recommend. The DES algorithm is the default algorithm when you install RACF on your system.

The DES algorithm is identified in the Federal Information Processing Standard 46-1 of the Computer Systems Laboratory in Gaithersburg, Maryland, of the National Institute of Standards and Technology of the United States Government. DES is accepted as a national and international standard.

The RACF DES algorithm

Encryption programs in general imply a two-way process: encryption and decryption.

- Encryption is a process that uses an encryption key and the data itself as inputs. The result is an encrypted form of the data.
- Decryption reverses the process; that is, the encrypted form of the data can only be decrypted by using the encryption key and the encrypted form of the data as inputs to reverse the encryption process.

The RACF DES authentication algorithm provides a high level of security because it supports one-way encryption only; *it does not support the reverse process.* In addition, it does not store the password it uses as the encryption key. For these reasons, the reconstruction of original data is virtually impossible. However, make sure that users do not have READ access to the RACF database unless their jobs require it.

How the RACF DES algorithm works

When a user changes a password, RACF treats the new, user-supplied password (or OIDCARD data) as an encryption key to transform the RACF user ID into an encoded form, using the DES algorithm, that it stores on the database. The password is not stored.

When a user logs on and enters a password or OIDCARD, RACF encrypts the user ID using the DES algorithm, using the password (or OIDCARD data) as the key. RACF then compares the results with the encoded form stored on the database using the DES compare function. If they match, then the password is valid.

Migrating to the RACF DES authentication algorithm

When an installation moves from the masking algorithm to the RACF DES algorithm, many users' passwords on the system might already be encoded through the masking algorithm.

The two-step method of password authentication: To move easily from a masked form to a DES form of authentication, RACF provides a two-step method of password authentication. The two-step method of password authentication is used when RACF cannot find an ICHDEX01 exit in the link pack area.

The two-step method causes a gradual migration from the masking algorithm to the RACF DES algorithm. The migration works as follows:

- When a user whose password has been encoded using the masking algorithm changes the password, RACF uses the DES algorithm to transform the RACF user ID into an encoded form that it stores on the RACF database.
- Each time the user logs on and enters a password or OIDCARD, RACF performs the two-step method of authentication as follows:
  1. RACF first compares the results of the DES algorithm to the encoded form of the password stored on the database. If the password cannot be verified, then the second step is performed.
  2. RACF compares the results of the masking algorithm to the encoded form of the password stored on the database.

Notes:

1. If two or more systems share the RACF database, they must all use the same password authentication algorithm.
2. For RACF Version 1, the masking algorithm is the default password authentication algorithm. Therefore, if a RACF Version 2, OS/390, or z/OS system is sharing the database with a RACF Version 1 system, both systems cannot use their default authentication algorithms. You must ensure that all the systems use the same authentication algorithm.

   If you do not ensure that the systems use the same algorithm, RACF might not be able to recognize valid passwords, and users might not be able to log on. For example, systems A and B are both at level 1.9.2 and share a database. They use the masking algorithm, which is the default. System A is updated to RACF Version 2, and now uses the default DES algorithm. If a user changes her password on the Version 2 system, RACF uses the DES algorithm to store an encoded form of the user ID on the RACF database. If the same user then tries to log on to the 1.9.2 system, RACF uses the masking algorithm to verify her password, and the password is not recognized.
3. If you use an installation application or add-on product that passes or synchronizes encrypted or masked password data between two RACF databases, you should ensure that all systems using the databases are using the same algorithm.
4. You can use the RACF remote sharing facility to synchronize passwords between RACF databases, even if the systems using the databases do not use the same password authentication algorithm. However, this is not recommended because a network is only as secure as its weakest point of entry. We recommend that you use the DES authentication algorithm on all systems in an RRSF network, to reduce the risk of compromising a password that can be used on multiple systems.

For further information on ICHDEX01, see “Password authentication exits” on page 275.

Using the DES algorithm without the two-step method of checking

Your installation might wish to use the DES algorithm without using the two-step method of checking. For example, if your installation is installing RACF for the first time and has never used the masking algorithm, or if all of your users’ passwords have been RACF DES-encoded, you do not need the two-step method.

There is an extremely remote possibility that DES-encrypting a user ID with the real password could give the same result as masking the user ID with a different password, allowing a password that is not valid to be accepted. As long as your installation uses the two-step method of checking, your installation may have an

exposure. You can minimize this possibility by using the DES algorithm without the two-step method of checking if you do not need to check for masked passwords.

To use the DES algorithm without the two-step method of checking, write an ICHDEX01 exit (in the link pack area) that sets the return code to 8. See “Password authentication exits” on page 275.

Using the masking algorithm

If your installation has been using the masking algorithm, you might want to continue to use it after installing RACF Version 2 or the Security Server. This is not recommended, however, because the DES algorithm provides better security. To do this, activate the ICHDEX01 exit that is shipped with RACF in SYS1.LINKLIB, or write your own ICHDEX01 exit (in the link pack area) that sets the return code to 4. For more information, see “Password authentication exits” on page 275.

In addition, you must provide an ICHDEX11 exit that performs function equivalent to the ICHDEX01 exit. Write your own ICHDEX11 exit (in the link pack area) that sets the return code to 4. See “Password authentication exits” on page 275.

Using your own authentication algorithm

Your installation might wish to use your own algorithm for authenticating passwords, instead of one of the algorithms provided by RACF. To do this, write an ICHDEX01 exit and an ICHDEX11 exit (in the link pack area) to perform your authentication algorithm, and set the return code to 0. See “Password authentication exits” on page 275.

PassTicket authentication

The RACF secured signon function provides an alternative to the RACF password called a *PassTicket*. Instead of having the user’s clear text password flow over the network, a RACF PassTicket can be generated by a requesting product or function, and used as the user’s authenticator to a RACF secured network application. In addition to the possibility of improved security for passwords within the network, PassTicket technology can be used to effectively move the authentication of a mainframe application user ID from RACF to another authorized function running on the host system, or to the work station local area network (LAN) environment. If RACF authenticates a password field and determines that it is not the RACF password for the user ID, RACF may perform a second authentication step to determine whether the password field is a valid PassTicket. See “How RACF processes the password or PassTicket” for more information. See *z/OS Security Server RACF Macros and Interfaces* for information on generating PassTickets.

How RACF processes the password or PassTicket

To validate a password or PassTicket, RACF:

1. Determines whether the value in the password field is the RACF password for the user ID.
   - If it is the RACF password, the validation is complete.
   - If it is not the RACF password, processing continues.
2. Determines whether a secured signon application profile has been defined for the application in the PTKTDATA class.
   - If a profile has not been defined, RACF sends a message to the user ID indicating that the password is not valid.
   - If the application is defined to the PTKTDATA class, processing continues.

3. Evaluates the value entered in the password field. The evaluation determines whether:
   - The value is a PassTicket consistent with this user ID, application, and time range.
   - When PassTicket replay protection is in effect (replay protection is not being bypassed), RACF checks to be sure the PassTicket has not been used previously on this computer system for this user ID, application, and time range.

   Note: A PassTicket is considered to be within the valid time range when the time of generation (with respect to the clock on the generating computer) is within plus or minus 10 minutes of the time of evaluation (with respect to the clock on the evaluating computer).

   If the value is determined to be a valid PassTicket, the user is allowed access to the desired application. If the value is not a valid PassTicket, RACF sends a message indicating that the user entered a password that is not valid.
4. Gives the user ID access to the desired application if the PassTicket is valid.

Notes:

1. For RACF to properly evaluate PassTickets, the TOD clock must be properly set to Greenwich Mean Time (GMT) rather than local time. (GMT is also referred to as coordinated universal time (UTC).)
2. If the RACF secured signon application key is encrypted, the cryptographic product must be active when RACF tries to authenticate the PassTicket. If it is not active, RACF cannot validate the PassTicket. The resulting message indicates that the logon attempt failed.
3. If the evaluation fails, the host application sends the user a message stating that the value in the password field is not valid.

Changing the RACF report writer options (ICHRSMFI module)

The RACF report writer provides a wide range of management reports that enable your installation to assess system and resource use. The report writer lists information contained in the SMF records that RACF generates. The RACF report writer can do the following:

- List the contents of RACF SMF records in a format that is easy to read.
- Produce reports that describe attempts to access a particular RACF-protected resource. These reports contain the user ID, number and type of successful accesses, and number and type of unauthorized access attempts.
- Produce reports that describe user and group activity.
- Produce reports that summarize system use and resource use.

For more information on the RACF report writer and the RACF report-writer command (RACFRW), see *z/OS Security Server RACF Auditor's Guide*.

ICHRSMFI is an installation-replaceable, non-executable module that contains default values for the SORT and MERGE parameters, the dynamic-allocation parameters, and the processing options used by the RACF report writer.

Table 2. Format of ICHRSMFI

Name	Offsets DEC(HEX)	Length	Description	Format	Default
SORTMAIN	0(0)	3	SORT/MERGE main-storage value	EBCDIC (MAX) or binary. Zero means ignore this parameter.	0
SORTRSRV	3(3)	3	SORT/MERGE reserved main-storage value	Binary. Zero means ignore this parameter.	0
SORTMSG	6(6)	3	SORT/MERGE message option	EBCDIC (NOF, (U), or (I))	(U)
SORTDDNM	9(9)	8	SORT/MERGE ddname for messages	EBCDIC, left-justified, and padded with blanks	SYSOUT
SORTTECH	17(11)	4	SORT/MERGE sorting technique	EBCDIC (PEER, BALN, OSCL, POLY, CRCX, or all blanks). Blanks mean selected by SORT/MERGE.	PEER
SORTTBL	21(15)	256	SORT/MERGE alternate sequence distribution table	Binary	No table provided
SORTTBLS	277(115)	2	SORT/MERGE alternate-sequence distribution-table size. (This parameter equals the actual number of non-blank characters in the SORTTBL parameter field.)	Binary (0 or 256). Zero means ignore this table.	0
SORTDYN	279(117)	32	SORT/MERGE dynamic allocation of intermediate workspace parameter	EBCDIC and left-justified	DYNALLOC= SYSDA
SORTDYNS	311(137)	2	SORT/MERGE dynamic-allocation parameter size. (This parameter equals the actual number of non-blank characters in the SORTDYN parameter field.)	Binary. Zero means no dynamic allocation by SORT/MERGE.	14
SORTEQU	313(139)	8	SORT/MERGE preservation of input order for records with equal sort fields	EBCDIC (EQUALS or NOEQUALS)	NOEQUALS
SORTEQUS	321(141)	2	SORT/MERGE SORTEQU field size. (This parameter equals the actual number of non-blank characters in the SORTEQU parameter field.)	Binary (6 for EQUALS and 8 for NOEQUALS)	8
SORTDSN	323(143)	44	SORT/MERGE SORTLIB data set name. May be blanks if no SORTLIB is needed.	EBCDIC, left-justified, and padded with blanks	SYS1.SORTLIB
OUTSPA1	367(16F)	2	SYSPRINT primary-space allocation (in tracks)	Binary	50
OUTSPA2	369(171)	2	SYSPRINT secondary-space allocation (in tracks)	Binary	20

Table 2. Format of ICHRSMFI (continued)

Name	Offsets DEC(HEX)	Length	Description	Format	Default
OUTBLKSI	371(173)	2	SYSPRINT block size	Binary	3192
OUTCLASS	373(175)	1	SYSPRINT output class	EBCDIC (A-Z or 0-9)	A
WRKSPA1	374(176)	2	SORTIN primary-space allocation (in tracks)	Binary	50
WRKSPA2	376(178)	2	SORTIN secondary-space allocation (in tracks)	Binary	20
WRKLRECL	378(17A)	2	SORTIN logical-record size	Binary	8192
WRKBLKSI	380(17C)	2	SORTIN block size	Binary	8196
WRKUNIT	382(17E)	8	SORTIN unit	EBCDIC, left-justified, and padded with blanks. All blanks mean information is obtained from Protected Step Control Block.	SYSDA
WRKSER	390(186)	6	SORTIN volume serial	EBCDIC, left-justified, and padded with blanks. All blanks mean no specific volume serial.	All blanks
INBUFFSI	396(18C)	4	Size of internal buffer for rebuilding SMF records	Binary	2048
INITREC	400(190)	1	SMF record type used for job initiation / TSO logon recording	Binary	20

For additional information, see *z/OS DFSORT Application Programming Guide*.

You should review the defaults in ICHRSMFI to ensure that they apply to your current operating environment.

To change the ICHRSMFI default values, construct an SMP/E USERMOD with ++ZAP statements to add the new values to the ICHRSMFI module.

Note: If you reinstall RACF, be sure to reapply these changes.

Customizing the RACF remote sharing facility

For information on customizing the RACF remote sharing facility, see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119.

Customizing the RACF/DB2 external security module

For information on customizing the RACF/DB2 external security module, see Chapter 6, “The RACF/DB2 external security module,” on page 187.

Chapter 4. Operating considerations

Enabling and disabling RACF 62
Enabling RACF 63
Disabling RACF 63
Dynamic parse 64
Syntax of the IRRDPI00 command 65
RACF authorization of the IRRDPI00 command 65
TSO/E authorization 65
Automating IRRDPI00 65
Running IRRDPI00 from the RACF parameter library 65
Running IRRDPI00 from a started procedure 66
ACEEs and VLF considerations 67
Dependencies 67
Operation 68
Removing information from VLF 68
Shared database considerations 69
VLF considerations for mapping UIDs and GIDs. 69
Dependencies 69
VLF considerations for caching user security packets (USPs) 69
Dependencies 69
The RACF subsystem 69
Activating the RACF subsystem. 70
Updating the IEFSSNxx member of SYS1.PARMLIB 71
Assigning a user ID to the RACF subsystem 74
The RACF PROC 75
Restarting the RACF subsystem 76
Restarting a function in the RACF subsystem 76
Examples 77
Restarting a function after applying maintenance 77
Restarting a function to recover from failures 78
Stopping the RACF subsystem address space 78
Diagnosing problems in the RACF subsystem 79
RACF operator commands 79
Group tree in storage 79
Shared database considerations 80
Using the global resource serialization function 80
RACF ENQ resources 81
Using Application Identity Mapping. 83
Sysplex considerations 84
Sharing a database 85
Sharing a database with sysplex communication in non–data sharing mode 86
Sharing a database with sysplex communication in data sharing mode 86
Sysplex communication. 87
Non–data sharing mode 88
Data sharing mode 88
Read-only mode 89
Failsoft mode 89
Enabling sysplex communication 89
Inactive backup data sets 90
Defining RACF structures for the coupling facility 90
System authorization facility (SAF). 93
The SAF router. 93
The SAF callable services router 94
Associating started procedures and jobs with user IDs 94

© Copyright IBM Corp. 1994, 2004

Methods for associating started procedures with RACF identities 96
The STARTED class 97
The started procedures table (ICHRIN03) 98
Coding the started procedures module 98
Generic entry in ICHRIN03 99
Changing the ICHAUTAB module 102
Failsoft processing 102
General considerations 103
Impact on users 104
CICS considerations 105
CICS timeout value 105
DFP considerations 105
TSO considerations 106
ISPF considerations 106
DB2 considerations 106
DASD data sets 106
Using utilities on RACF-protected DASD data sets 107
Using utilities with the OPERATIONS or group-OPERATIONS attribute 108
Renaming RACF-protected data sets 108
Using IEHMOVE with the ADSP attribute 109
Using IEHMOVE with the COPYAUTH parameter 110
Using the DFSMSdss and DSF utilities 110
Moving a RACF-indicated DASD data set between systems 110
Moving a RACF-indicated data set to a RACF-active system 111
Moving a data set with a discrete profile to a RACF-inactive system 111
Moving a RACF-indicated data set to a non-RACF system with RACF indicator checking 111
Moving a multivolume RACF-indicated data set between systems 112
Using access method services commands 113
LISTCAT command 113
REPRO/RESETCAT/IMPORT/IMPORTRA commands 113
DASD volumes 113
Scratching DASD data sets 114
Moving DASD volumes between systems 114
UCBs above 16MB 114
Protecting tape data 114
Tape data protection and bypass label processing (BLP) 114
Considerations for unlabeled (NL) tapes 115
Using utilities on RACF-protected tape volumes and tape data sets 115
Moving tape volumes between systems 115
Moving multivolume tape data sets between systems 115
Multiple users per address space 116
Restarting jobs 116
Panel driver interface 116
REXX RACVAR function 117
Installing the REXX RACVAR function 117
Using the REXX RACVAR function 117

This chapter describes aspects of certain functions that you should consider when you operate a system that has RACF installed.

Enabling and disabling RACF

RACF is enabled and disabled on by entries in the IFAPRD*xx* member of SYS1.PARMLIB.

Enabling RACF

Before you can use RACF on z/OS, it must be enabled. At install time, an entry must exist in the IFAPRD*xx* member pointed to by PROD= in IEASYS*xx* (in SYS1.PARMLIB) to enable RACF. If a correct entry does not exist, RACF initialization does not complete, IFA104I is issued, and RACF offers no security for the system.

If you order RACF as part of the Security Server for z/OS, the IFAPRD*xx* entry should look like this:

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('Security Server')
        ID(5694-A01)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(ENABLED)
```

If you order the Security Server but want to use a security product other than RACF, there should be two IFAPRD*xx* entries that look like this:

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('Security Server')
        ID(5694-A01)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(ENABLED)

PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('RACF')
        ID(5694-A01)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(DISABLED)
```

If you need to make any changes to your IFAPRD*xx* parameter library member, see *z/OS MVS Product Management*. After you make your changes, re-IPL the system to make the changes take effect. The SET PROD=*xx* command does not affect RACF's enablement state; only the status of the IFAPRD*xx* member at IPL time affects RACF's enablement state.

Disabling RACF

To disable RACF, update the appropriate IFAPRD*xx* member and change the STATE field to:

```
STATE(DISABLED)
```

Then re-IPL the system to make the change take effect.

For example, if you ordered RACF as part of the Security Server for z/OS, and you want to disable the Security Server, update the IFAPRD*xx* entry to look like this:

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('Security Server')
        ID(5694-A01)
```

```
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(DISABLED)
```

If you ordered RACF as part of the Security Server for z/OS, and want to disable the RACF component of the Security Server but continue to use the other components (for example, DCE), update the IFAPRD*xx* entries to look like this:

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('Security Server')
        ID(5694-A01)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(ENABLED)

PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('RACF')
        ID(5694-A01)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(DISABLED)
```

Dynamic parse

Dynamic parse is required to parse segment-related keywords. It is used to add, list, alter, or delete DFP, TSO, or any other non-base segment information with the RACF commands.

Commands that refer only to RACF base information will work, provided they do not contain any typing mistakes. If you make a mistake, dynamic parse will parse the command and issue a message for an entry that is not valid (the typing error).

All RACF installations must run IRRDPI00 at IPL time. Otherwise, command processors that allow segment information will not work, even if segment information is not being used.

Attention

Do not change the dynamic-parse specification data.

If IBM makes updates to the dynamic-parse specification data (IRRDPSDS) and the database templates (IRRTEMP2), you must apply the new templates to the RACF database and activate them on each system *before* activating dynamic parse or entering the IRRDPI00 command. To apply the new templates to the database, run IRRMIN00 with PARM=UPDATE from one system that uses the database. To activate the templates on a system, run IRRMIN00 with PARM=ACTIVATE, or re-IPL. The PTF containing the update will specify whether or not you should update your templates. For information on IRRMIN00, see Chapter 7, “RACF database utilities,” on page 201. After you update your templates, run IRRDPI00 on each system if the PTF also changed the dynamic parse specifications. Each time IRRDPI00 runs, RACF remembers the level of the IRRDPSDS data set. Each time you IPL, RACF activates the latest level of the RACF templates on the system. You can display the level of each that your system is using with the RACF operator command SET LIST. The level is an FMID, such as HRF2220, or an APAR number.

For the templates, the level also includes an 8-digit release level, and an 8-digit APAR level. Figure 15 on page 154 shows a sample of the SET LIST output.

RACF command processors require the dynamic parse table for processing commands. This table is built by the IRRDPI00 command.

Syntax of the IRRDPI00 command

```
IRRDPI00     {CHECK | LIST | UPDATE}
```

where:

CHECK
: Performs syntax checks on the input data set (SYSUT1 DD statement)

LIST
: Lists the current contents of the dynamic parse table

UPDATE
: Performs syntax checks on the input data set and updates the dynamic parse table if no errors were found

RACF authorization of the IRRDPI00 command

To invoke the IRRDPI00 command, you must be one of the following:

- Permitted to IRRDPI00 in the FACILITY class

```
RDEFINE FACILITY IRRDPI00 UACC(NONE)
PERMIT IRRDPI00 CLASS(FACILITY) ID(xxxxxxxx) ACCESS(READ)
```

- Permitted to IRRDPI00 using RACF program control

```
RDEFINE PROGRAM IRRDPI00 -
     ADDMEM(SYS1.LINKLIB/SYSRES/NOPADCHK) UACC(NONE)
PERMIT IRRDPI00 CLASS(PROGRAM) ID(xxxxxxxx) ACCESS(READ)
```

- Defined as a RACF SPECIAL user

TSO/E authorization

IRRDPI00 must be added to the TSO/E APF-authorized command table.

If you are using the SYS1.PARMLIB member IKJTSO*xx* to define the APF-authorized RACF commands and programs, update this member to include IRRDPI00. A sample, IKJTSOxx, is provided in member RACPARM of SYS1.SAMPLIB.

If you are not using the SYS1.PARMLIB member IKJTSO*xx*, you must modify two TSO/E CSECTs. See *z/OS TSO/E Customization* for information on how to do this.

Automating IRRDPI00

Because you must run IRRDPI00 after every IPL, IBM recommends that you automate it. There are two methods you can use to automate IRRDPI00:

- Run IRRDPI00 from a RACF parameter library member that automatically runs at RACF subsystem initialization.
- Run IRRDPI00 from a started procedure that automatically runs at every IPL.

Running IRRDPI00 from the RACF parameter library

To run IRRDPI00 from the RACF parameter library, do the following:

1. If you already have a RACF parameter library member set up to run automatically when the RACF subsystem initializes, determine which member it

is. Otherwise, create a new IRROPT*xx* member and set it up to run automatically when the RACF subsystem initializes. See “The RACF parameter library” on page 168 for information.

2. Add the following commands to the beginning of the IRROPT*xx* member, ahead of any TARGET commands or any other RACF commands that affect profile segments other than the base segment:

```
ALLOCATE FILE(SYSUT1) DATASET('SYS1.SAMPLIB(IRRDPSDS)') SHR
IRRDPI00 UPDATE
FREE     FILE(SYSUT1)
```

3. Make sure that IRRDPI00 is in the TSO/E APF-authorized command table. See “TSO/E authorization” on page 65 for more information.
4. If you want to test your changes, you can execute the commands in the IRROPT*xx* parameter library member by issuing the SET command (for example, `#SET INCLUDE(xx)`). Or, you can restart the RACF subsystem to execute them. An IPL will also execute them, but is usually not desirable.
5. If you previously ran IRRDPI00 from a started procedure such as IRRDPTAB, you should remove the started procedure. The removal should include:
   - Removing the started procedure from SYS1.PROCLIB
   - Removing the `COM='START IRRDPTAB'` statement from the appropriate COMMND*xx* member
   - Removing an entry specifically for the IRRDPTAB started procedure in the started procedures table (ICHRIN03) or the STARTED class, if you set one up

Running IRRDPI00 from a started procedure

You can set up PARMLIB and PROCLIB to automatically invoke the IRRDPTAB started procedure, which issues the IRRDPI00 UPDATE command, after every IPL. To do this:

1. Add the IRRDPTAB started procedure to SYS1.PROCLIB. This creates a started task that executes the TSO terminal monitor program in batch and issues the IRRDPI00 UPDATE command. Here is a sample procedure that is contained in SYS1.SAMPLIB member RACPROC:

```
//IRRDPTAB PROC
//*
//*THIS STARTED TASK IS RUN AT IPL TO LOAD THE RACF
//*DYNAMIC PARSE TABLES.  THE USERID FOR THE TASK
//*MUST BE AUTHORIZED TO ISSUE THE IRRDPI00 COMMAND.
//*
//         EXEC PGM=IKJEFT01,REGION=1M,
//              PARM='IRRDPI00 UPDATE'
//SYSTSPRT DD  SYSOUT=Z,HOLD=YES
//SYSUDUMP DD  SYSOUT=Z,HOLD=YES
//SYSUT1   DD  DSN=SYS1.SAMPLIB(IRRDPSDS),DISP=SHR
//SYSTSIN  DD  DUMMY
```

2. Create or update the COMMND*xx* PARMLIB member to include a start command for the IRRDPTAB procedure. Once created, the COMMND*xx* PARMLIB member should be added to an IEASYS*xx* member to ensure that the command is invoked after each IPL.

```
COM='START IRRDPTAB'
```

   Refer to *z/OS MVS Initialization and Tuning Guide* for more details on the coding of the COMMND*xx* PARMLIB member.

3. Assign a RACF user ID and group name to the IRRDPTAB started procedure using either the started procedures table (ICHRIN03) or the STARTED class. See “Associating started procedures and jobs with user IDs” on page 94 for more information.
4. Authorize the RACF user ID associated with the IRRDPI00 started procedure. See “RACF authorization of the IRRDPI00 command” on page 65 for details.
5. Add IRRDPI00 to the TSO/E APF-authorized command table. See “TSO/E authorization” on page 65 for more information.

ACEEs and VLF considerations

RACF can save ACEEs (accessor environment elements) using VLF (virtual lookaside facility) and retrieve them for later use. If you have multiple requests to RACF to build a user security environment (ACEE), you can benefit. You might see improvements in areas such as logon, batch job submissions, MVS/APPC, and CICS reconnect. The amount of improvement is related to how often RACF finds the necessary data in VLF.

Dependencies

In order for this function to be available, VLF must be active.

Update the COFVLF*xx* member of SYS1.PARMLIB as follows:

```
CLASS NAME(IRRACEE)        /* RACF ACEE Data in Memory
 */
EMAJ (ACEE)                /* Major name = ACEE              */
```

Before activating the IRRACEE class, check for installation use of the ACEEIEP field. Nonstandard use of the field requires the IRRACX01 and IRRACX02 installation exits. See “ACEE compression/expansion exits” on page 255. RACF storage of ACEEs in VLF might affect the processing of your pre- and postprocessing exits, because the ACEE passed to these exits might have been retrieved from VLF. Therefore it might have already been modified by your exits when the ACEE was originally created. The area pointed to by ACEEIEP is retrieved with the ACEE. Before reusing ACEEIEP, installation code must process any existing area pointed to by it. A pointer to storage might be lost if installation code stores over ACEEIEP.

Once the IRRACEE class of VLF objects is active, invokers automatically receive the benefits of saving ACEEs in VLF. If the class is not active, requests normally using the function shift to using the RACF database.

The default amount of storage supplied with a VLF class should be adequate. However, if you have users connected to many groups, or if you have a large number of users, or if you have attached your own information off the ACEE, you might need to increase storage. RACF stores one object in VLF for each ACEE, but that object can hold several copies of ACEEs for that user. RACF keeps a separate ACEE for each combination of:

- Group name
- Port of entry (POE), for example, terminal ID or console ID
- Application name
- Security label
- Session type

If a user logged on from three separate terminals, for example, three ACEEs would be saved in the user's VLF object. If the user also ran a batch job, a fourth ACEE would be saved. Thus, the VLF objects could be much bigger than the expected size of a single ACEE.

Notes:

1. The IRRACEE class must not be defined to VLF unless all RACF systems sharing the database (MVS and VM) are using RACF 1.9.2 or later.
2. If you have an MVS or VM system at the RACF 1.9 level sharing the database, you should *not* define the IRRACEE class. This would cause out-of-date ACEEs to be used on the higher-level systems.
3. Define the VLF class when all sharing systems (MVS and VM) are at the RACF 1.9.2 level or later. If this is not possible and you must share the RACF database between a system at the RACF 1.9 level and a system at the RACF 1.9.2 level or higher, RACF administration tasks must be performed from the higher level system. If the RACF database is shared between systems at the RACF 1.9.2 level or higher, RACF administration tasks can be performed from either system.

Operation

VLF is searched before the RACF database to see if the ACEE for a particular user exists. If it does, this data is used in building the security environment, to avoid I/O to the RACF database. If there is no entry, RACF builds the ACEE entry as it normally would, but saves the ACEE for later use. If the ACEE had to be built by RACF, it is not stored in VLF until after ICHRIX02 (the RACROUTE REQUEST=VERIFY postprocessing exit) has been invoked.

Removing information from VLF

RACF monitors security-related changes to ensure that the information in VLF is valid. RACF removes the ACEE of the particular user from VLF if it determines that a security-related change has occurred.

A security-related change is:

- Removing a user from a particular group.
- Changing a "security-sensitive" field in a user's security profile. Security-sensitive fields can be identified by referring to the RACF database templates in *z/OS Security Server RACF Macros and Interfaces*. A security-sensitive field has bit 0 of flag 2 turned on.

The commands that make security-related changes are those that manipulate user profiles (for example, ALTUSER, DELUSER, and ADDUSER).

For security-related changes where all of the incorrect user ACEE entries cannot be determined, all the ACEEs will be removed from VLF. Examples of these changes are defining entire groups or updates from another system (VM or MVS) sharing the database.

Issuing commands that deal with certain general-resource classes can cause information to be removed from VLF. The classes are:

- APPCPORT
- APPL
- CONSOLE
- FACILITY, when the SETROPTS MLS option is active
- GTERMINL

- JESINPUT
- SECLABEL
- SERVAUTH
- TERMINAL

Whenever the RACF SETROPTS command specifies the CLASSACT, NOCLASSACT, RACLIST REFRESH, or NORACLIST keywords for one of these classes, RACF considers all of the ACEEs in VLF to lack integrity, and removes them from VLF. ACEE saving continues as the ACEEs are subsequently rebuilt.

Shared database considerations

If systems share a RACF database and are not running in sysplex communication or data sharing mode, when RACF removes one or more ACEEs from VLF on one of the sharing systems, the other systems do not know which ACEEs were removed. In these cases, RACF removes *all* of the ACEEs from VLF on *all* of the sharing systems.

VLF considerations for mapping UIDs and GIDs

With application identity mapping stage 0, 1, or 2, RACF can use VLF to map z/OS UNIX user identifiers (UIDs) to user IDs and z/OS UNIX group identifiers (GIDs) to group names for verification of z/OS UNIX System Services requests. For more information on performance considerations, see “Mapping UIDs to user IDs and GIDs to group names” on page 34.

Dependencies

In order for this function to be available, VLF must be active and the active COFVLF*xx* member of SYS1.PARMLIB must include statements defining the VLF classes used for the mapping. Update the COFVLF*xx* member of SYS1.PARMLIB as follows:

```
CLASS NAME(IRRGMAP)        /* GMAP table for z/OS UNIX System Services */
EMAJ (GMAP)                /* Major name = GMAP                         */
CLASS NAME(IRRUMAP)        /* UMAP table for z/OS UNIX System Services */
EMAJ (UMAP)                /* Major name = UMAP                         */
```

VLF considerations for caching user security packets (USPs)

RACF can use VLF to cache user security packets (USPs), in order to improve the performance of z/OS UNIX System Services applications that use thread level security services.

Dependencies

For this function to be available, VLF must be active and the active COFVLF*xx* member of SYS1.PARMLIB must include statements defining the IRRSMAP VLF class. Update the COFVLF*xx* member of SYS1.PARMLIB as follows:

```
CLASS NAME(IRRSMAP)        /* SMAP table for z/OS UNIX System Services */
EMAJ (SMAP)                /* Major name = SMAP                         */
```

The RACF subsystem

The RACF subsystem enables remote RACF administration and password synchronization, provides an execution environment for most RACF commands and provides support for APPC persistent verification. Starting the subsystem is optional but recommended. It is not necessary for system IPL or most RACF functions, but it is required for the following functions:

- RACF remote sharing facility

The RACF subsystem is required for the RACF remote sharing facility to be operational. For more information see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119.

- RACF commands as operator commands

  When the RACF subsystem is active, most RACF commands can be issued as operator commands. For more information, see “RACF operator commands” on page 79.

- R_admin (IRRSEQ00) callable service

  When the RACF subsystem is active, it executes commands which are passed to it by R_admin. Applications that use R_admin, such as TME 10™ User Administration, require the RACF subsystem to be active.

- RACF LU6.2 persistent verification

  The RACF subsystem provides a centralized data owner/data server environment for the signed-on lists used by RACF persistent verification. The lists are managed with the RACROUTE REQUEST=SIGNON macro. RACF also provides an execution environment for the RACF persistent verification operator commands, DISPLAY and SIGNOFF.

The RACF subsystem address space is identified as a standard MVS subsystem. The RACF subsystem provides the following services:

- Automatic start of the RACF subsystem at IPL time.
- Tailorability through startup parameters. The RACF subsystem reads startup parameters from the IEFSSN*xx* member of SYS1.PARMLIB and the PARM keyword on the EXEC statement in the subsystem procedure.
- Optional subsystem command identifiers. You can choose to use the MVS subsystem convention of assigning a unique subsystem prefix or you can use the unique subsystem name, followed by a blank, as the prefix for the RACF subsystem.

  This unique subsystem name is defined in the IEFSSN*xx* member of SYS1.PARMLIB.

Only one RACF subsystem can run at a time. If you define more than one RACF subsystem with the same name in IEFSSN*xx*, only one starts. It is possible to define two RACF subsystems with different names, and start the second one after stopping the first, but this is not recommended. If you choose to do this, you must specify PARM=INITIAL on the MVS START command whenever you start a RACF subsystem that has a different name than the one that was previously running.

Activating the RACF subsystem

To activate the RACF subsystem, you must:

- Update the IEFSSN*xx* member of SYS1.PARMLIB
- Update the SCHED*xx* member of SYS1.PARMLIB
- Assign a RACF user ID to the RACF subsystem
- Review the RACF PROC provided in SYS1.PROCLIB

  The RACF remote sharing facility requires a SYSLBC card to access the broadcast data set. When the ADDUSER or ALTUSER command adds or modifies a TSO segment, the broadcast data set is updated.

Figure 6 on page 71 shows the relationship between the IEASYS*xx* member of SYS1.PARMLIB and the IEFSSN*xx* member in establishing:

- The name of the RACF subsystem, and

- The name of the RACF procedure in SYS1.PROCLIB that is executed when the MVS START command is issued

The name of the subsystem specified in the IEFSSN*xx* member and the name of the RACF procedure must be identical.

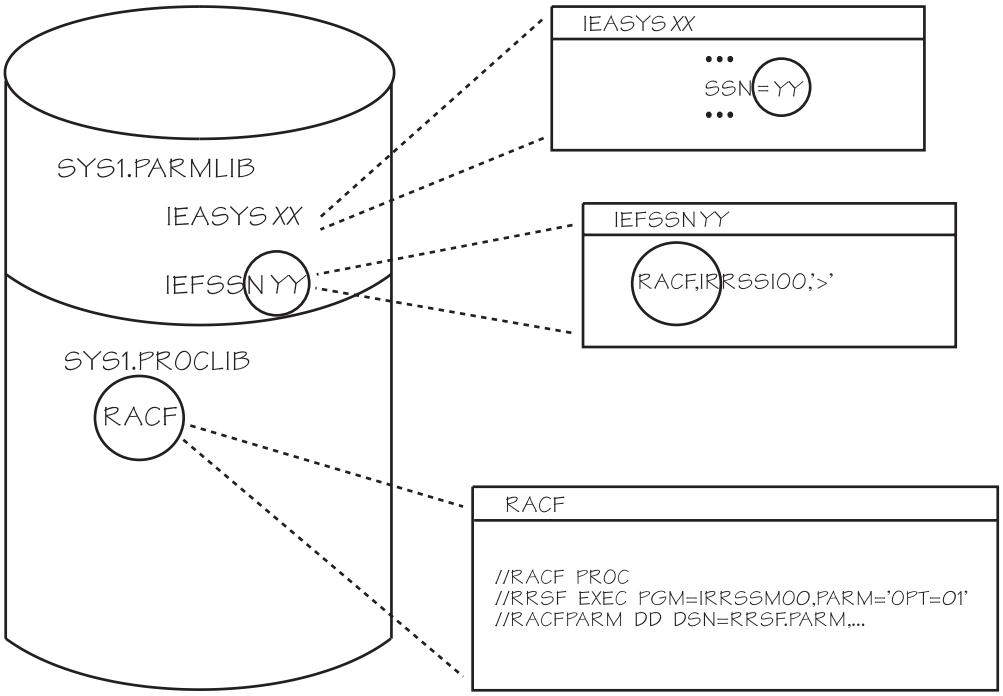


Figure 6. Setting Up the RACF Subsystem Address Space. In this example, the subsystem name is RACF. The operator specifies the xx for IEASYSxx at IPL time via the SYSP=xx parameter.

Updating the IEFSSNxx member of SYS1.PARMLIB

The IEFSSN*xx* member must be updated to indicate that the RACF subsystem is a valid subsystem in the installation. This member also identifies the subsystem's command prefix, used in issuing RACF operator commands, and an optional command prefix scope. See *z/OS Security Server RACF Command Language Reference* for information on how to use the subsystem command prefix.

You can choose to have RACF register the command prefix with the MVS command prefix facility (CPF). CPF ensures that two or more subsystems do not have the same or overlapping command prefixes. CPF also allows an operator or authorized application to enter a RACF command from any system in a sysplex and route that command to run on another system in the sysplex. The command responses come back to the originating system console. For more information on CPF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

We recommend that you have RACF register command prefixes with CPF. To do this, specify a scope on the IEFSSN*xx* entry.

You can code the IEFSSN*xx* definition in a keyword parameter form or a positional parameter form. The keyword parameter form has the following syntax:

```
SUBSYS SUBNAME(ssname) INITROUTINE(IRRSSI00) [INITPARM('cmdpref[,scope]')]
```

and the positional parameter form has the following syntax:

```
ssname,IRRSSI00[,'cmdpref[,scope]']
```

where:

```
ssname    is the 1-4 character subsystem name (required)
IRRSSI00  is the RACF subsystem initialization routine (required)
cmdpref   is the 1-8 character command prefix (optional)
scope     is the command prefix scope for CPF (optional)
   X                for sysplex scope
   M                for system scope
```

We recommend that you use the keyword parameter form. Subsystems defined using the keyword parameter form of the IEFSSN*xx* parmlib member can use dynamic SSI services, while subsystems defined using the positional form of the IEFSSN*xx* parmlib member cannot use dynamic SSI services. For information about dynamic SSI services, see *z/OS MVS Using the Subsystem Interface*. For information about coding the IEFSSN*xx* parmlib member, see *z/OS MVS Initialization and Tuning Reference*.

If you do not specify a command prefix, the default is the subsystem name plus a blank, and the command prefix is not registered with CPF. Messages from the subsystem display the subsystem name, enclosed in parentheses, instead of a command prefix.

If you do not specify a scope, the quotes around the command prefix are optional.

Do not define a command prefix that is the same as an existing command prefix on that system. Do not define a command prefix that is a subset of, or a superset of, an existing command prefix on that system with the same first character. For example, if command prefix $ABC exists, $, $A, and $AB are subsets of $ABC and conflict with it. $ABCD is a superset of $ABC and conflicts with it. You can define command prefix ABC, however, because it does not start with the same letter as $ABC and so does not conflict. You can see which prefixes already exist using the DISPLAY OPDATA command. See *z/OS MVS System Commands* for information on the DISPLAY OPDATA command.

If you do not specify a scope, the command prefix is not registered with CPF. We recommend that you specify a scope. If you specify sysplex scope, the command prefix must be unique within the sysplex, and a command with the prefix can be issued from another system in the sysplex to run on the system identified by the command prefix. If you specify system scope, the command prefix must be unique within the system, and a command with the prefix runs on the system on which it is issued (or to which it is routed via the MVS ROUTE command).

If the registration with CPF fails (for example, if the command prefix is already registered with CPF), the subsystem is unavailable. Restart the subsystem to make it available (see “Restarting the RACF subsystem” on page 76). The restarted subsystem uses the default command prefix (the subsystem name) and the prefix is not registered with CPF. Messages from the subsystem display the subsystem name, enclosed in parentheses, instead of a command prefix. If you correct the IEFSSN*xx* member, you must re-IPL for the change to take effect.

Examples of IEFSSN*xx* entries

- If the entry in IEFSSN*xx* is:

  ```
  SUBSYS SUBNAME(RACF) INITRTN(IRRSSI00)
  ```

  or

  ```
  RACF,IRRSSI00
  ```

RACF is the subsystem name and 'RACF ' (the subsystem name followed by a blank) is the command prefix by default. Because no scope is specified, the command prefix is not registered with CPF.

- If the installation assigns a unique subsystem identifier and the entry in IEFSSN*xx* is:

  ```
  SUBSYS SUBNAME(RACF) INITRTN(IRRSSI00) INITPARM('#')
  ```

  or

  ```
  RACF,IRRSSI00,'#'
  ```

  RACF is the subsystem name and # is the command prefix. Because no scope is specified, the command prefix is not registered with CPF.
- If the entry in IEFSSN*xx* is:

  ```
  SUBSYS SUBNAME(RACF) INITRTN(IRRSSI00) INITPARM('%')
  ```

  or

  ```
  RACF,IRRSSI00,%
  ```

  RACF is the subsystem name and % is the command prefix. Because no scope is specified, quotes are optional on the command prefix.
- If the entry in IEFSSN*xx* is:

  ```
  SUBSYS SUBNAME(RACF) INITRTN(IRRSSI00) INITPARM('#RACF1')
  ```

  or

  ```
  RACF,IRRSSI00,'#RACF1'
  ```

  RACF is the subsystem name and #RACF1 is the command prefix. Because no scope is specified, the command prefix is not registered with CPF.
- If the entry in IEFSSN*xx* is:

  ```
  SUBSYS SUBNAME(RACF) INITRTN(IRRSSI00) INITPARM('#,M')
  ```

  or

  ```
  RACF,IRRSSI00,'#,M'
  ```

  RACF is the subsystem name and # is the command prefix. The prefix has system scope, so a command with this prefix runs on the system on which it is entered (or to which it is routed via the MVS ROUTE command). Because a scope is specified, the command prefix is registered with CPF.
- If the entry in IEFSSN*xx* is:

  ```
  SUBSYS SUBNAME(RAC3) INITRTN(IRRSSI00) INITPARM('%RACSS1,X')
  ```

  or

  ```
  RAC3,IRRSSI00,'%RACSS1,X'
  ```

  RAC3 is the subsystem name and %RACSS1 is the command prefix. The prefix has sysplex scope, so a command with this prefix runs on this system no matter where it is issued within the sysplex. Because a scope is specified, the command prefix is registered with CPF.

Notes:

1. For JES2 systems, if a command prefix is specified in the IEFSSN*xx* member, it must differ from any BSPACE= value defined in the CONDEF part of the JES HASPARM definition.

2. The command prefix precedes the message ID for some subsystem messages. If you choose to use a long prefix, you should consider the appearance of the subsystem messages as well as the usability of typing a long prefix on an operator command. Consider including a separator character such as a hyphen at the end of a long prefix to separate it from subsystem message IDs.

Assigning a user ID to the RACF subsystem

The RACF subsystem must have a valid RACF user ID. The RACF subsystem cannot be initialized if a valid RACF user ID is not assigned to it. The PROC name for the RACF subsystem must be the same as the name used in IEFSSN*xx*.

We recommend that you assign a *protected user ID* to the RACF subsystem. A user ID becomes a protected user ID when it is assigned both the NOPASSWORD and NOOIDCARD attributes by an ADDUSER or ALTUSER command. A protected user ID cannot be revoked due to incorrect password attempts or used to enter the system in ways that require a password. For information on protected user IDs, see *z/OS Security Server RACF Security Administrator's Guide*.

In a remote sharing environment, the first seven characters of the user ID assigned to the RACF subsystem are displayed at the end of TSO XMIT messages after a command is successfully directed. You might want to consider this when you choose the RACF subsystem user ID.

You can assign a RACF user ID to the RACF subsystem using the started procedures table (ICHRIN03), as shown in the following example, or using the STARTED class.

Note: The following example is not really representative of ICHRIN03 because it has only one entry.

```
ICHRIN03   CSECT
NUMBER     DC     X'8001'        Number of entries in started procedures table
PROC       DC     CL8'RACF    '  Name of the RACF subsystem
USERID     DC     CL8'RACFAS  '  Name of RACF-defined user ID
GROUP      DC     CL8'        '
FLAGS      DC     X'00'
RESERVED   DC     XL7'00000000000000'
           END
```

It is usually not necessary to mark the entry *privileged* or *trusted*. However, if you are using remote sharing functions you should mark the entry as privileged or trusted so that the RACF subsystem has access to data sets used for remote sharing functions, including:

- Workspace data sets
- APPC-related profiles
- The RACF parameter library

The following example illustrates how to assign a RACF user ID to the RACF subsystem and mark it privileged:

```
ICHRIN03   CSECT
NUMBER     DC     X'8001'        Number of entries in started procedures table
PROC       DC     CL8'RACF    '  Name of the RACF subsystem
USERID     DC     CL8'SUBSYS  '  Name of RACF-defined user ID
GROUP      DC     CL8'        '
FLAGS      DC     X'80'          Entry is privileged
RESERVED   DC     XL7'00000000000000'
           END
```

For more information on remote sharing, see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119.

If your installation has activated the STARTED class, instead of updating the started procedures table you can create a profile for the RACF subsystem in the STARTED class, as shown in the following example:

```
RDEFINE STARTED RACF.* STDATA( USER(RACFAS) PRIVILEGED(YES) )
```

where RACF is the name of the subsystem and RACFAS is the RACF-defined user ID.

See “Associating started procedures and jobs with user IDs” on page 94 for more information.

The RACF PROC

Sample JCL to activate the RACF subsystem is provided in SYS1.PROCLIB. You can replace it with your own procedure, but be sure the name of the procedure matches your IEFSSN*xx* entry. Here is an example:

```
//RACF  PROC
//RACF  EXEC PGM=IRRSSM00,REGION=0M
```

The JCL *must* specify IRRSSM00 as the name of the module on the EXEC JCL statement.

If there is an entry in the IEFSSN*xx* member for the RACF subsystem, and the subsystem name matches the name of the RACF procedure, the RACF procedure is started automatically at IPL time.

If you want to use a RACF parameter library, your JCL must include a RACFPARM DD statement specifying the name of the RACF parameter library partitioned data set. If you want a member of the parameter library to be automatically processed during initialization, your JCL should also include the PARM='OPT=*xx*' parameter on the EXEC statement to specify which parameter library member you want processed.

If you do not specify a suffix, it defaults to `00`. If you include a RACFPARM DD statement, but do not include the PARM='OPT=*xx*' parameter on the EXEC statement, RACF assumes that you want a parameter library member processed automatically, and defaults to IRROPT00. If you include a RACFPARM DD statement in your JCL because you want to use a RACF parameter library, but you do *not* want a member processed automatically, be aware that:

- If you have an IRROPT00 member, RACF processes it automatically during initialization, even if you don’t specify it on the PARM='OPT=*xx*' parameter.
- If you don’t have an IRROPT00 member, and don’t specify a PARM='OPT=*xx*' parameter, RACF tries to find an IRROPT00 member during initialization, and issues a warning that it couldn’t find one.

The following JCL activates the RACF subsystem and automatically processes the member IRROPT01 in the RACF parameter library contained in data set RRSF.PARM:

```
//RACF     PROC
//RRSF     EXEC PGM=IRRSSM00,REGION=0M,PARM='OPT=01'
//RACFPARM DD DSN=RRSF.PARM,DISP=SHR
```

For more information on the RACF parameter library, see “The RACF parameter library” on page 168.

Restarting the RACF subsystem

When the RACF subsystem is active, it detects subsystem failures and attempts to restart itself. This processing occurs multiple times. If the subsystem is unable to restart successfully, it eventually terminates. (For a description of the information that can be lost when the RACF subsystem address space is not active, see "Stopping the RACF subsystem address space" on page 78.)

If the subsystem terminates, after you resolve the error condition you can restart the subsystem manually in one of two ways. The first method you should try is to enter the following command at the MVS operator's console:

```
START xxxx,SUB=MSTR
```

where *xxxx* is the subsystem name chosen by the installation. Of the two restart methods, this method is the least destructive to current work executing in the subsystem.

If your attempt to restart the subsystem with the preceding command fails, it might still be possible to restart the RACF subsystem by entering the following command at the MVS operator's console:

```
START xxxx,SUB=MSTR,PARM=INITIAL
```

where *xxxx* is the subsystem name chosen by the installation. This second form of the START command might be necessary in rare circumstances where residual data is incompatible and cannot be reused.

When you use command prefix registration (CPF), the RACF prefix is registered only once, during subsystem initialization at IPL. Restarting the RACF subsystem does not alter the command prefix registration. If you restart the RACF subsystem after a failed attempt to register the command prefix with CPF, the subsystem uses the default command prefix (the subsystem name) and the prefix is not registered with CPF.

Restarting a function in the RACF subsystem

Use the RESTART operator command to restart a specified function in the RACF subsystem. You can use the RESTART command to recover from failures when the RACF subsystem is unable to recover automatically. You can also use it to restart a subtask after applying maintenance, which can, in some cases, allow the maintenance to become effective without requiring that you stop and restart the entire RACF subsystem or re-IPL. Figure 7 shows the syntax of the RESTART command. See *z/OS Security Server RACF Command Language Reference* for more information on the RESTART command. See *z/OS Security Server RACF Security Administrator's Guide* for information on security for the RESTART command.

```
prefixRESTART  { COMMAND
               | CONNECTION [ NODE(nodename | *)
                  [ SYSNAME(sysname) | * ) ] ]
               | MESSAGE
               | OUTPUT
               | RACLINK
               | RECEIVE
               | SEND }
```

Figure 7. RESTART Command Syntax

The functions you can specify on the RESTART command are:

Function Keyword	Function Restarted	Modules Reloaded
COMMAND	Command and application update handler	IRRSSC00
CONNECTION	Device driver manager	IRRDDM00
	Device drivers	IRRAPPC0, IRRAPPC2
	Local node device driver	IRRSSL00
	Handshaking	IRRAPPC6
CONNECTION NODE	Device drivers	none
MESSAGE	Message processor	IRRSSMG0
OUTPUT	Output handler	IRRSSOP0
RACLINK	RACLINK task	IRRSSK00
RECEIVE	RRSF request receiver	IRRSSR00
SEND	RRSF request sender	IRRSSND0

The RESTART CONNECTION NODE (*nodename*) command can be used to restart a pair of device drivers. The NODE keyword specifies the remote node for which the device driver pairs are to be restarted. You can specify NODE(*) to restart all device drivers.

Examples

The command:

```
RESTART CONNECTION NODE(nodename)
```

restarts the connection to the node *nodename*, if *nodename* is a single-system RRSF node. If *nodename* is a multisystem node, RACF issues an error message and does not execute the command.

The command:

```
RESTART CONNECTION NODE(*)
```

or

```
RESTART CONNECTION NODE(*) SYSNAME(*)
```

restarts the connections to all single-system RRSF nodes and to all member systems of multisystem nodes.

The command:

```
RESTART CONNECTION NODE(nodename) SYSNAME(sysname)
```

restarts the connection to the specific member system *sysname* on the multisystem node *nodename*. If *nodename* is a single-system RRSF node, RACF issues an error message.

The command:

```
RESTART CONNECTION NODE(nodename) SYSNAME(*)
```

restarts the connections to all member systems of the multisystem node *nodename*. If *nodename* is a single-system RRSF node, RACF issues an error message.

Restarting a function after applying maintenance

You can use the RESTART command after applying maintenance to a module in one of the load modules associated with a restartable function. When you RESTART the associated function, a new updated copy of the load module is made

available, without the need to re-IPL or reinitialize the RACF subsystem address space. The exception is that when you specify the NODE keyword, no modules are reloaded.

Restarting a function to recover from failures

When failures occur, RACF attempts to restart the failing function automatically. However, in some situations RACF issues a message indicating a recovery action that you must take, and directs you to issue a RESTART command.

Stopping the RACF subsystem address space

You can use the RACF STOP operator command to stop the RACF subsystem address space. Use the RACF STOP command instead of the MVS FORCE command, which has some dangerous side effects, such as not cleaning up resources. Figure 8 shows the syntax of the RACF STOP command. See *z/OS Security Server RACF Command Language Reference* for more information on the RACF STOP command. See *z/OS Security Server RACF Security Administrator's Guide* for information on security for the RACF STOP command. (Note that there is also an MVS STOP command. The RACF STOP command and the MVS STOP command are not related.)

```
prefixSTOP
```

Figure 8. RACF STOP Command Syntax

Like other system address spaces that are needed for basic system operation, the RACF subsystem address space runs non-cancellable. If you are *not* using the RACF remote sharing facility, there is no need to stop the address space before system shutdown. Subsequent IPL or MVS START command processing rebuilds the address space for proper system usage.

If you *are* using the RACF remote sharing facility, the RACF STOP command allows you to stop the RACF subsystem address space with a minimal loss of remote requests. Stopping the RACF subsystem address space any other way (for example, using FORCE) is not recommended, and might cause requests to be lost or the VSAM files for workspace data sets to be damaged. If an RRSF node allows directed commands, password synchronization, or automatic direction, it is important that you stop the address space with the RACF STOP command during a system shutdown, after all users have logged off and all batch jobs have completed. It is also important that you *not* stop (or FORCE) the address space while users and jobs are active. If you do, updates could be lost, resulting in passwords or profiles not being synchronized, and output from directed commands could be lost.

The STOP command does the following:

- Breaks all connections to other nodes.
- Prevents RRSF requests waiting in the INMSG workspace data sets from being dispatched. These requests remain in the INMSG files and are executed when the RACF subsystem is started again.
- Rejects all requests to send additional work to the subsystem address space.
  - Any inbound APPC requests are rejected as node dormant.
  - All RACF TSO commands that use the address space (those with the AT or ONLYAT keyword specified) are rejected with an error message indicating that the subsystem is not active.

- Allows current outbound messages to complete, but no additional messages are sent. Outbound messages remain in the OUTMSG workspace data sets, and are sent to the remote nodes after the RACF subsystem is started again and the connections are reestablished.
- Stops any RRSF requests that remain running after five seconds, and leaves them in the workspace data sets.
- Saves the output from RRSF requests that have completed, for transmission after the address space is restarted.
- Closes and deallocates the VSAM files for the workspace data sets.
- Stops all remaining tasks in the address space.

If you stop the RACF subsystem address space on an RRSF node, password synchronization and automatic direction can no longer send RRSF requests to remote nodes, and updates that should be made on remote nodes are lost. RRSF requests directed to that node from remote nodes are queued in the workspace data sets at the remote nodes, and are not lost.

For example, assume that ADDUSER commands are being automatically directed between NODEA and NODEB, and the STOP command is issued on NODEA. If a user issues an ADDUSER command on NODEA, the command runs on NODEA, but is not directed to NODEB. However, if a user issues an ADDUSER command on NODEB, the command runs on NODEB, and is held on NODEB until communication with NODEA is re-established. The command is then directed to NODEA.

For information on remote sharing and automatic direction, see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119.

Diagnosing problems in the RACF subsystem

You can perform traces to diagnose possible problems in the RACF subsystem. For information on tracing, see *z/OS Security Server RACF Diagnosis Guide*.

RACF operator commands

Most RACF commands can run as operator commands in the RACF subsystem address space. You can control the ability to issue RACF commands as operator commands with OPERCMDS profiles. For more information on running RACF commands as operator commands, see *z/OS Security Server RACF Command Language Reference*. For information on the profiles needed to protect RACF commands when they run as operator commands, see *z/OS Security Server RACF Security Administrator's Guide*.

Group tree in storage

RACF can improve the performance of selected group attribute use by exploiting MVS/ESA hardware using the virtual lookaside facility (VLF) to create objects in data spaces. This approach reduces I/O to the RACF database when determining the structure of the group tree.

Group-tree-in-storage processing can shorten RACF processing during logon if a user needs group-OPERATIONS authority to access a data set during the logon process. If someone with group authorities is using those authorities (for example, issuing commands, accessing data sets through group operations), and that user logs on and off multiple times during the day, the installation should see a performance benefit with group-tree-in-storage processing.

Note: If the user's group-related activities are very light, less benefit will be seen.

Group-tree-in-storage processing also improves the time needed to run RACF commands when the user has group-SPECIAL, group-AUDITOR, or group-OPERATIONS authority. If the installation has created multiple users with group authorities and those users have control over some of the same groups, activating the group-tree-in-storage support should provide performance savings.

The default amount of storage supplied with a VLF class should be adequate.

To activate this support for group tree in storage, the class name IRRGTS must be described in the VLF COFVLF*xx* PARMLIB member. If IRRGTS is not described, then the support for group tree in storage is not active.

The activation or deactivation is *not* effective until the next VLF restart or the next IPL.

COFVLF*xx* PARMLIB member:

```
CLASS NAME(IRRGTS)     /* VLF class name for RACF GTS function */
      EMAJ(GTS)        /* Major name of IRRGTS class           */
```

Refer to *z/OS MVS Initialization and Tuning Guide* for more details on the coding of the COFVLF*xx* PARMLIB member.

Shared database considerations

When the RACF database is to be shared, the device on which the database resides must be configured as shared, or damage to the database is likely. Both primary and backup databases must be shared. For information on how to define a device as shared, see *z/OS HCD User's Guide*.

Except when operating in data sharing or read-only mode, RACF serializes access to a shared database through use of the hardware RESERVE/RELEASE capability. Depending on the work characteristics of your systems, this use of RESERVE/RELEASE might cause contention problems. An installation that is experiencing contention problems related to the RACF database can consider converting the RESERVEs. If an installation uses OS/390's global resource serialization function to convert RESERVES, all MVS systems that access the RACF database must be part of the same global resource serialization complex, and there can be no VM systems sharing the database.

When running in data sharing mode or read-only mode, RACF uses global ENQs instead of the hardware RESERVE/RELEASE capability. These ENQs should occur at a lower rate than the RESERVEs would occur. However, when running in non–data sharing mode, RACF uses hardware RESERVES to serialize database access, unless the installation has explicitly converted hardware RESERVEs to ENQs using OS/390's global resource serialization function.

Using the global resource serialization function

A global resource serialization complex lets users on multiple MVS systems serialize access to processing and logical resources, such as data sets on shared DASD volumes.

Using global resource serialization to convert hardware RESERVES to ENQs minimizes several problems:

- Interlocks
- Job contention for the same volume
- One system monopolizing a shared device
- Data integrity exposures occurring if a system resets while a RESERVE is in effect

To convert the RESERVEs, place a generic entry for SYSZRACF in the RESERVE conversion resource name list (RNL) in your GRSRNL*xx* member in SYS1.PARMLIB. For further information, see *z/OS MVS Planning: Global Resource Serialization*.

RACF ENQ resources

RACF uses the following ENQ names to serialize on resources. SYSTEMS type names must be propagated throughout the sysplex. SYSTEM or STEP type names must not be propagated beyond the local system.

Table 3. RACF ENQ Resources

Major	Minor	Type	Notes
ICHRGL01	SIGNON.ENQ	STEP	
SYSRACF2	SSTABLE1	SYSTEM	
SYSRACF2	SSTABLE2	SYSTEM	
SYSZRACF	*racfdsn*	SYSTEMS	*racfdsn* is the name of the RACF data set being serialized. In non–data sharing mode, RACF issues this as a RESERVE; however, your global resource serialization product, such as GRS, can be used to convert this to an ENQ, which must be treated as a multisystem ENQ. In data sharing mode, RACF issues this as an ENQ. Your global resource serialization product must treat this as a multisystem ENQ.
SYSZRACF	*subsys*.CMD.*cmd*	STEP	*subsys* is the RACF subsystem name, *cmd* is the name of the command being issued.
SYSZRACF	*subsys*|*member*	STEP	*subsys* is the RACF subsystem name. | means concatenation. *member* is the parmlib member being processed.
SYSZRACF	ACEE3PTY*@@@@*bbbb*	STEP	@@@@ is a hex address. *bbbb* is 4 blank characters.
SYSZRACF	ACEE3PTY*@@@@CGRP	STEP	@@@@ is a hex address.
SYSZRACF	AHSTABLE	STEP	
SYSZRACF	AHSTUSER*userid*	STEP	*userid* is the caller's user ID.
SYSZRACF	CNSTGNLP**classname*	SYSTEM	*classname* is the class being processed.
SYSZRACF	CNSTRCLP**classname*	SYSTEM	*classname* is the class being processed.
SYSZRACF	RACF	SYSTEM	
SYSZRACF	SETROPTS	SYSTEMS	

Table 3. RACF ENQ Resources (continued)

Major	Minor	Type	Notes
SYSZRACF	DSDTDSDT...DSDT	SYSTEM	The minor name consists of 12 occurrences of ″DSDT″ concatenated into one string. In non–data sharing mode, RACF issues this as a SYSTEM ENQ. Your global resource serialization product, such as GRS, must treat this as a single system ENQ.
SYSZRACF	DSDTDSDT...DSDT	SYSTEMS	The minor name consists of 12 occurrences of ″DSDT″ concatenated into one string. In data sharing mode, RACF issues this as a SYSTEMS ENQ. Your global resource serialization product, such as GRS, must treat this as a multisystem ENQ.
SYSZRACF	DSDTPREP...DSDTPREP	SYSTEMS	The minor name consists of 6 occurrences of ″DSDTPREP″ concatenated into one string.
SYSZRACP	*dsn*	SYSTEMS	*dsn* is any data set name. RACF issues this as a RESERVE; however, your global resource serialization product, such as GRS, can be used to convert this to an ENQ, which must be treated as a multisystem ENQ.
SYSZRAC2	*racfdsn*	SYSTEM	*racfdsn* is the name of the RACF data set being serialized.
SYSZRAC2	DPDTABPT@@@@	SYSTEM	@@@@ is a hex address.
SYSZRAC2	ICHSEC00	SYSTEM	
SYSZRAC2	IRRCRV05	SYSTEM	In non–data sharing mode, RACF issues this as a SYSTEM ENQ. Your global resource serialization product, such as GRS, must treat this as a single system ENQ.
SYSZRAC2	IRRCRV05	SYSTEMS	In data sharing mode, RACF issues this as a SYSTEMS ENQ. Your global resource serialization product, such as GRS, must treat this as a multisystem ENQ.
SYSZRAC2	IRRDPI08@@@@	SYSTEM	@@@@ is a hex address.
SYSZRAC2	RACGLIST_*classname*	SYSTEMS	*classname* is the class being processed.
SYSZRAC2	RCVTDPTB@@@@	SYSTEM	@@@@ is a hex address.
SYSZRAC2	SMCFIX	STEP	
SYSZRAC2	XMCAXMCA...XMCA	SYSTEM	The minor name consists of 12 occurrences of ″XMCA″ concatenated into one string.
SYSZRAC2	GLOBALGLOBALGLOBAL	SYSTEMS	

Table 3. RACF ENQ Resources (continued)

Major	Minor	Type	Notes
SYSZRAC2	PROGRAMPROGRAMPROGRAM	SYSTEMS	
SYSZRAC2	CONNECT...CONNECT	SYSTEM	The minor name consists of 6 occurrences of ″CONNECT″ concatenated into one string.
SYSZRAC2	CACHECLS_*cachename*	SYSTEMS	*cachename* is the name of an R_cacheserv managed cache
SYSZRAC2	TEMPLATE	SYSTEM	Serializes the activation of a new set of in-storage templates.
SYSZRAC3	*rrsfdsn*	STEP	*rrsfdsn* is the RRSF data set output name.
SYSZRAC4	*index_entry*	SYSTEMS	*index_entry* is the alias index entry name.
SYSZRAC4	*template* / *profile*	SYSTEMS	*template* / *profile* is the database template number concatenated with the base profile name.
SYSZRAC4	U*n*/G*n*	SYSTEMS	The minor name consists of U*n* or G*n* where *n* is a valid UID or GID value.
SYSZRAC4	BPX.NEXT.USER	SYSTEMS	Obtained to serialize updates to the BPX.NEXT.USER profile in the FACILITY class.
SYSZRAC5	ALIAS	SYSTEMS	
SYSZRAC5	IRRIRA00	SYSTEMS	
SYSZRAC8	00*cachenamebbbbbbbbbb*DASPHYTR	SYSTEM	*cachename* is the name of an R_cacheserv managed cache. *bbbbbbbbbb* is 10 blank characters.
SYSZRAC8	00*cachenamebbbbbbbbbb*DASPHYST	SYSTEM	*cachename* is the name of an R_cacheserv managed cache. *bbbbbbbbbb* is 10 blank characters.
SYSZRAC8	00*cachenamebbbbbbbbbb*CACHE	SYSTEM	*cachename* is the name of an R_cacheserv managed cache. *bbbbbbbbbb* is 10 blank characters.
SYSZRAC8	00*cachenamebbbbbbbbbb*DASPHYDS	SYSTEM	*cachename* is the name of an R_cacheserv managed cache. *bbbbbbbbbb* is 10 blank characters.

Using Application Identity Mapping

With a database created at OS/390 Release 10 or higher, RACF's application identity mapping uses an alias index to associate user and group names with Lotus Notes for z/OS, Novell Directory Services for OS/390, and z/OS UNIX identities. RACF typically uses mapping profiles for this purpose with databases created before Release 10. However, you can use the IRRIRA00 utility to convert the database to work with an alias index instead. The conversion consists of a series of stage transitions from zero to three. RACF uses the database mapping information differently for each stage. For a complete description of the stages, see "RACF internal reorganization of aliases utility program (IRRIRA00)" on page 203.

Systems with an older release are not aware of an alias index or the or the application identity mapping conversion stages, so you should use care when sharing a database between older systems and a system at Release 10 or higher.

- If you share a RACF database at stage 0, all systems use mapping profiles. There is nothing to you need to do.
- If you share a RACF database at stage 1 or higher, you can negatively affect the mapping information by issuing a USER or GROUP command from a system running a release earlier than OS/390 Release 10.

Example:

In your sysplex, System 1, at Release 10 level, shares a RACF database with system 2, at release 8 level, and the RACF database is at stage 1.

- If you add a mapping, for example, a UID, on system 1, the alias index structures and the mapping profiles remain synchronized.
- If you add a mapping on system 2, which can use only mapping profiles, the alias index structure gets out of sync on the RACF database.

To prevent this from happening, make your changes from system 1 only. To prevent changes being made from system 2 inadvertently, do one of the following:

- Use the program control with a system identifier to protect the commands from being issued on system 2. The steps are described in *z/OS Security Server RACF Security Administrator's Guide*.
- Do not migrate your RACF database to stage 1 or higher until all systems sharing the database are at Release 10 or higher.

Sysplex considerations

The MVS sysplex is an evolving platform for the large system computing environment. It offers you improved price/performance through cost-effective processor technology and enhanced software. It increases system availability and your potential for doing more work.

A major difference between a sysplex and a conventional large computer system is the improved growth potential and level of availability in a sysplex. The sysplex increases the number of processing units and MVS operating systems that can cooperate, which in turn increases the amount of work that can be processed.

Because work can be distributed around the sysplex, all systems in the sysplex must have the same security information. Therefore, we recommend that all systems in a sysplex use the same RACF database. Definitions for security categories (members of the CATEGORY profile in the SECDATA general resource class) are likely to cause problems if all systems do not use the same RACF database. For more information on security categories, see *z/OS Security Server RACF Security Administrator's Guide*.

The *coupling facility* allows MVS and other software to share data concurrently among multiple systems in the sysplex with the goal of maintaining a single system image.

A sysplex with a coupling facility significantly changes the way systems can share data. The technology that makes high performance sysplex data sharing possible is a combination of hardware and software services. *Data sharing* is the ability of concurrent subsystems or application programs to directly access and change the same data while maintaining system integrity.

The goal of RACF in the sysplex is to provide security for the resources of all systems in a comprehensive and centralized way. RACF allows you to use the coupling facility and shared RACF data to help manage the security of resources for all systems in a sysplex.

The following are valuable for learning more about a sysplex and the coupling facility.

- *z/OS Security Server RACF Diagnosis Guide*
- *z/OS Parallel Sysplex Overview*
- *z/OS MVS Setting Up a Sysplex*
- http://www.s390.ibm.com/s390/pso/psotool

Sharing a database

See “Shared database considerations” on page 80 for important considerations that apply even within a sysplex.

RACF is designed so that its database can be shared between processor complexes while data integrity is maintained. Because of the need to serialize RACF database processing, there might be some I/O contention. RACF sysplex data sharing is designed to address the problems that can occur when many systems share a RACF database. Sharing the RACF database requires that:

- The database reside on shared DASD.
- The data set name table (ICHRDSNT) is compatible on all sharing systems.
- The database range table (ICHRRNG) is identical on all sharing systems.
- The class descriptor table (ICHRRCDE) is compatible on all sharing systems.

Systems can share a RACF database if they have any of the following versions of RACF installed:

- RACF Version 1 on MVS, using the restructured database format
- RACF Version 2 on MVS
- RACF on VM
- The RACF component of the OS/390 Security Server
- The RACF component of the z/OS Security Server

However, you must not share outside the sysplex if you choose to use data sharing mode (see “Sharing a database with sysplex communication in data sharing mode” on page 86), nor outside of a single global resource serialization complex if you choose to convert the SYSZRACF RESERVE to an ENQ using the OS/390 global resource serialization reserve conversion RNL.

Many SETROPTS options (for example SETROPTS CLASSACT) automatically take effect across sharing systems when the inventory control block (ICB) of the master primary RACF data set is read on the other systems. Other SETROPTS commands (for example SETROPTS RACLIST) must be issued explicitly on all sharing systems. RACF sysplex communication introduces command propagation for certain commands (for example SETROPTS RACLIST). You no longer have to issue these commands on each system sharing the database. Instead, you can issue a command once, and RACF propagates it to the other systems in the sysplex. For more information on command propagation, see “Sysplex communication” on page 87.

Sharing a database with sysplex communication in non–data sharing mode

Before you enable your RACF system for sysplex communication in non–data sharing mode, the system *must* meet the following requirements:

- MVS release 4.2 or higher is running.
- RACF release 2.1 or higher is running.
- The system must be a single-system sysplex or a member of a multisystem sysplex (that is, not in XCF-local mode).
- If you are using OS/390's global resource serialization function to serialize system resources, the major names SYSZRACF and SYSZRAC2 cannot be in the exclusion resource name list (RNL).

  If you have one of these names in your RNL, you must schedule a sysplex-wide IPL to remove it before running RACF in sysplex communication or datasharing mode, or your RACF database might become corrupted. You cannot remove these names dynamically, because RACF maintains a permanent ENQ on some of these resources.
- If you are using a non-IBM global resource serialization product to serialize system resources, be aware that resources with major names SYSZRACF and SYSZRAC2 might be requested with SCOPE=SYSTEMS. You must ensure that SCOPE=SYSTEMS is honored for the requests.
- As noted in "Shared database considerations" on page 80, in non–data sharing mode hardware RESERVEs are used to serialize database access. If using OS/390's global resource serialization function, you could consider converting the RESERVEs to ENQs by placing a generic entry for SYSZRACF in the RESERVE conversion resource name list (RNL). For more information, see "Using the global resource serialization function" on page 80.
- The system and all systems it shares a database with meet the requirements for sharing a database:
  - The database resides on shared DASD.
  - The data set name table (ICHRDSNT) is compatible on all sharing systems.
  - The database range table (ICHRRNG) is identical on all sharing systems.
  - The class descriptor table (ICHRRCDE) is compatible on all sharing systems.

We recommend that all systems enabled for sysplex communication and sharing the same RACF database be members of the same sysplex. Doing this prepares you for RACF sysplex data sharing, and ensures that commands are propagated to all members of the sysplex.

Sharing between sysplex members and systems outside the sysplex:
Systems enabled for sysplex communication and running in non–data sharing mode *can* share a database with other RACF systems which are not enabled for sysplex communication. Systems that are not enabled for sysplex communication but that are sharing the database cannot exploit command propagation. In this mixed environment, the systems must not enter data sharing mode. Moreover, because the systems within the sysplex cannot be in the same OS/390 global resource serialization complex as the systems outside the sysplex, you must not replace hardware RESERVEs with ENQs. This means you must *not* place a generic entry for SYSZRACF in the RESERVE conversion resource name list (RNL). As is the case when all systems are members of the same sysplex, the major names SYSZRACF and SYSZRAC2 cannot be in the exclusion resource name list (RNL).

Sharing a database with sysplex communication in data sharing mode

When RACF enters data sharing mode, all members of the sysplex change modes at the same time. **If you want to use data sharing mode, all systems sharing**

the database must be members of the same sysplex. Before RACF can enter data sharing mode, the following additional requirements must be met:

- *All* sharing systems are enabled for sysplex communication.
- *All* sharing systems are running MVS release 5.1 or later.
- *All* sharing systems have RACF 2.1 or later installed, OS/390 Security Server RACF installed and enabled, or z/OS Security Server RACF installed and enabled.

  Note: If an OS/390 Release 2 or higher system does not have RACF enabled, it does not join the data sharing group, but other systems are not affected and can still enter the data sharing group.
- *All* sharing systems have access to the same coupling facilities.
- RACF structures are defined to the coupling facility policy.
- All systems must not be in XCF-local mode.
- If you are using the global resource serialization function to serialize system resources, the major names SYSZRACF and SYSZRAC2 cannot be in the exclusion resource name list (RNL).

  If you have one of these names in your RNL, you must schedule a sysplex-wide IPL to remove it before running RACF in sysplex communication or datasharing mode, or your RACF database might become corrupted. You cannot remove these names dynamically, because RACF maintains a permanent ENQ on some of these resources.
- If you are using a non-IBM global resource serialization product to serialize system resources, be aware that resources with major names SYSZRACF and SYSZRAC2 might be requested with SCOPE=SYSTEMS. You must ensure that SCOPE=SYSTEMS is honored for these requests.
- All sharing systems meet the requirements for sharing a database:
  - The database resides on shared DASD.
  - The data set name table (ICHRDSNT) is compatible on all sharing systems.
  - The database range table (ICHRRNG) is identical on all sharing systems.
  - The class descriptor table (ICHRRCDE) is compatible on all sharing systems.

Attention

If RACF is in data sharing mode, all systems sharing the RACF database must be in the same sysplex. If you attempt to share the database with an MVS, OS/390, or z/OS system that is outside the sysplex, or with a VM system, database corruption will occur.

If you have RACF 2.2, OS/390, or z/OS systems that need to use the same security data, but are not all members of the same sysplex, you can give a system outside of a sysplex its own copy of the RACF database used by the sysplex, and use automatic direction to keep the databases synchronized. See “Automatic direction” on page 123 for information on automatic direction.

Sysplex communication

If you are in the planning stages of configuring for a sysplex, go to the IBM Internet site at `http://www.ibm.com/s390/pso/psotool` for more information.

When RACF is enabled for sysplex communication, it uses XCF to join the RACF data sharing group, IRRXCF00. The data sharing group facilitates communication between systems in the sysplex enabled for sysplex communication. There is only one data sharing group per sysplex.

When RACF is enabled for sysplex communication, you have the ability to enter certain commands that now affect the whole sysplex, simplifying security management. The command is entered once and it propagates through the rest of the sysplex. These commands are:

- RVARY SWITCH
- RVARY ACTIVE
- RVARY INACTIVE
- RVARY DATASHARE
- RVARY NODATASHARE
- SETROPTS RACLIST (*classname*)
- SETROPTS RACLIST (*classname*) REFRESH
- SETROPTS NORACLIST (*classname*)
- SETROPTS GLOBAL (*classname*)
- SETROPTS GLOBAL (*classname*) REFRESH
- SETROPTS GENERIC (*classname*) REFRESH
- SETROPTS WHEN(PROGRAM)
- SETROPTS WHEN(PROGRAM) REFRESH

When RACF is enabled for sysplex communication, it allocates in-storage buffers for the backup database (20% of the number allocated for the primary database), to reduce I/O to the backup device. Additionally, a minimum of 50 buffers are used for the primary database. As a consequence, you might notice an increased ECSA usage for the in-storage buffers.

When RACF is enabled for sysplex communication, the system is in one of three modes: non–data sharing mode, data sharing mode, or read-only mode. (These modes are independent of the RRSF modes, local mode and remote mode. A system that is in a data sharing group in data sharing mode, for example, can at the same time also be an RRSF node in either local or remote mode.)

Non–data sharing mode

In non–data sharing mode RACF uses RESERVE/RELEASE serialization protocols. The coupling facility is not used. The database can be shared with systems running RACF 1.9 or RACF 1.9.2 running with either MVS or VM, RACF 1.10, or the Security Server for OS/390 or z/OS.

Data sharing mode

In data sharing mode, the coupling facility is exploited to provide a large buffer for records from the RACF database, allowing a decrease in the I/O rate to the RACF database.

To facilitate data sharing, RACF uses a serialization protocol that replaces RESERVE/RELEASE when the coupling facility is in use. This protocol uses GLOBAL enqueues to protect the integrity of RACF's data.

Note: The use of GLOBAL enqueues in place of RESERVE/RELEASE might require changes to your serialization product (OS/390 global resource serialization or its equivalent).

Within the coupling facility, storage is dynamically partitioned into *structures*: cache, list, or lock. RACF uses *cache* structures as high-speed buffers for storing shared

data with common read/write access. This high-speed buffer is used with the local system buffer to reduce I/O to the RACF database. It permits RACF to determine more easily if another system has made changes that invalidate records in the local buffer.

When RACF enters data sharing mode, the RACF data sharing address space, RACFDS, starts automatically. The address space remains up for the life of the IPL. It is not a started procedure, and so does not make use of attributes in the RACF started procedures table (ICHRIN03)or the STARTED class.

The RACFDS address space is started with a high dispatching priority to assure that the services it performs are completed in a timely way relative to other system activity.

Using the coupling facility with a single MVS image: It is possible to use the coupling facility with a single MVS image. In this case the database is not shared, but the volume on which the database resides must be configured as shared, and the system cannot be in XCF-local mode. Even with a single-system sysplex, you can use the coupling facility to reduce RACF database I/O.

Read-only mode

A system enters this emergency mode when use of the coupling facility is specified, but an error has made the coupling facility either inaccessible to or unusable by RACF.

A serialization mode compatible with data sharing mode is used, allowing other systems in the sysplex to be in data sharing mode. However, RACF database updates are not allowed by a system in read-only mode. When MVS notifies RACF that the condition has been resolved, RACF re-enters data sharing mode. If the condition cannot be resolved, the data sharing group can be switched to non–data sharing mode with the RVARY NODATASHARE command.

Any functional updates to the RACF database other than for statistical purposes fail for a system in read-only mode.

Failsoft mode

A system enters this emergency mode when the DSNT specifies that the system should be enabled for sysplex communication at IPL time, but this is not possible due to environmental conditions. For more information see “Failsoft processing” on page 102.

Attention: You should be aware that certain sysplex recovery scenarios might require you to bring up a member in XCF-local mode. This causes the system to enter RACF failsoft mode. You will be unable to log onto any TSO ID to proceed with recovery unless you follow the recommendations in “Sysplex recovery scenarios that require XCF-local mode” on page 314 and “Emergency data set name tables” on page 40.

Enabling sysplex communication

A flag in the RACF data set name table (ICHRDSNT) enables the system for sysplex communication. When enabled for sysplex communication, a RACF system can be in one of several modes for accessing the RACF database. The mode is set by another flag in the data set name table. The mode determines whether or not RACF is to use the coupling facility. See “The data set name table” on page 37.

The first system to complete RACF initialization establishes the data set name table. This table remains in effect until a sysplex-wide IPL occurs or an RVARY command is issued. Subsequent systems validate their data set name table against that of the data set name table previously established. If they are compatible, the local data set name table is used. If RACF data set names do not match exactly, or backup data set option flags do not match exactly, RACF issues a message and uses the table previously established. If a mode discrepancy is detected, RACF does not issue a message but simply overrides the mode to match the table previously established. We recommend that you use a common data set name table. However, if you want to have differing numbers of in-storage buffers on different systems, you must use multiple data set name tables.

Attention: The above sharing of the data set name table from the first IPLed member to subsequently IPLed members only occurs if subsequent members use a data set name table that specifies at least sysplex communications mode (that is, the one-byte flag field is set to one of the following:

- X'xxxx10xx'B
- X'xxxx01xx'B
- X'xxxx11xx'B

This is fully described in “RACF sysplex communication” on page 39.

The current mode can be modified without an IPL by using the RVARY DATASHARE or RVARY NODATASHARE command. Note, however, that RVARY cannot change the bit setting in your data set name table. Therefore, you should change your data set name table mode bit so RACF comes up in the mode you want if you ever have to do a sysplex-wide IPL.

Inactive backup data sets

In general, when a system joins a sysplex RACF becomes active only when all data sets in the primary database and their backups are successfully allocated and opened. However, there is one exception: if a backup data set is inactivated (by the RVARY command) before a system's attempt to join the sysplex, the system successfully joins as a member of the sysplex with RACF active. RACF marks the backup data set inactive and deallocated.

An RVARY with the LIST option shows information similar to the following for an inactive, deallocated backup data set called RACFDB.BACK1 and a successfully allocated and opened primary data set called RACFDB.PRIM1:

```
      IRRA011I (@) OUTPUT FROM RVARY:
ICH15013I RACF DATABASE STATUS:
ACTIVE   USE    NUMBER   VOLUME     DATASET
------   ---    ------   ------     -------
 YES     PRIM      1     D79PK4     RACFDB.PRIM1
 NO      BACK      1     *DEALLOC   RACFDB.BACK1
```

After the system joins the sysplex, a subsequent RVARY to activate RACFDB.BACK1 causes RACF to allocate and open RACFDB.BACK1 on all systems in the sysplex.

Defining RACF structures for the coupling facility

To use RACF data sharing you must reserve space in the coupling facility by defining cache structures in the coupling facility resource manager (CFRM) policy. You need one cache structure for each data set specified in your data set name table (ICHRNDST). For example, if you have one primary data set and one backup data set, you need to define two cache structures. When defining structures you must provide structure name and sizes.

Structure names: You need to provide a structure name for each structure you define in the CFRM policy. The format of RACF cache structure names is:

`IRRXCF00_ayyy`

where:

a is P for primary or B for backup

yyy is the relative position of the data set in the data set name table (a decimal number, 001-090)

Structure size: You need to provide a structure size for each structure you define in the CFRM policy. There is no standard structure size for RACF—you need to determine the structure size that is best for your environment. Among the factors that determine structure size are:

- The number of systems in the sysplex
- The number of local buffers
- The number of blocks in the data set the structure is for
- The system workload

The structure size you specify in the CFRM policy is the sum of the amount of storage needed by RACF and the amount of storage needed for coupling facility control information. To determine the amount of storage needed for coupling facility control information, see *PR/SM™ Planning Guide*, GA22-7236 (the newest version) or GA22-7123 (an older version), for the formula, and use the following characteristics of RACF cache structures:

- The target directory-to-data ratio is 1/1.
- The adjunct assignment indicator is 0.
- The data area element characteristic is 4.
- The maximum data area size is 1.
- The maximum number of storage classes is 1.
- The maximum number of castout classes is 1.

The following information helps you to determine the amount of storage needed by RACF.

When discussing structure sizes, there are four different structures sizes to consider: minimum, maximum, initial, and optimum.

- *Minimum* is the least amount of coupling facility storage RACF needs to successfully connect to the structure.
- *Maximum* is the amount of coupling facility storage it would take to contain the entire data set that the structure is for.
- *Initial* is an estimate of a size that allows you to do data sharing with acceptable performance.
- *Optimum* is the smallest structure size at which system performance is “good”. (If you are operating at a less than optimal structure size, an increase in the structure size significantly improves the I/O rate to the RACF database.)

Attention

RACF does not support the ALTER function of coupling facility structures. Therefore, do not specify the INITSIZE operand in the STRUCTURE statement. If you do, the size of the structure is limited to the INITSIZE value instead of the SIZE value, and if the INITSIZE value is less than the SIZE value, RACF issues an informational message, IRRX012I.

Minimum Structure Size: For each structure, RACF needs 4K of coupling facility storage for each local buffer. For example, if you have requested 255 local buffers in ICHRDSNT, RACF needs 255 × 4K = 1020K of coupling facility storage for the primary structure. You need to add to this value the amount of storage needed for coupling facility control information. The minimum structure size of the backup data set is 20% of the minimum structure size of the primary. (This is because the number of local buffers for a backup data set is 20% of the number specified for the primary data set.) If the number of local buffers specified varies from one system to another, you must use the largest value specified. The minimum establishes a lower bound, and your structure size should be larger.

Maximum structure size: The largest structure size you define should not exceed the size of the data set the structure is for. For example, if the data set has 5000 4K blocks, your maximum structure size is 5000 × 4K = 20 000K. The maximum establishes an upper bound, and your structure size should be smaller.

Initial Size: Use the following formula to calculate your initial coupling facility structure size:

```
structure size = (b × 4K) + (b/10 × 4K × n) + C
```

where:

b	=	the number of I/O buffers defined in ICHRDSNT. (For backup data sets use 20% of the value specified for the primary).
4K	=	the size of the buffer
b/10	=	10% of the total number of buffers per system, to allow different data reference patterns
n	=	the number of CPCs in a sysplex
C	=	the amount of storage required for coupling facility control information

For a 5-way sysplex with 255 I/O buffers, the primary structure size would be:

```
(255 × 4K) + (26 × 4K × 5) + C = 1540K + C
```

and the backup structure size would be:

```
(51 × 4K) + (5 × 4K × 5) + C = 304K + C
```

Optimum size: The initial size allows you to enter RACF data sharing with acceptable performance. Once you have done that, you might want to tune your system for optimal performance. Here are some things to consider as you tune your structure size.

It is often the case that a relatively small percentage of the profiles in a database account for a relatively high percentage of the database I/O which can occur when granting access to protected resources. This I/O can be reduced if space is provided in the coupling facility for these “high activity” profiles. (The reduction pertains to non-RACLISTed profiles only, because RACLISTed profiles do not require database I/O.) The reduction occurs because RACF looks in the coupling facility before going to the database. The situation is analogous to the well-known notion of working set, which has been used to describe the number of real storage page frames required to run a program without “thrashing”.

Providing coupling facility space beyond what is needed for the “high activity” profiles is less beneficial. For example, a data block containing a profile which is used only once requires I/O to bring it into the coupling facility. It is never referenced again, so nothing is gained.

One strategy for determining the optimum structure size is to monitor I/O rates to the RACF database and gradually increase the size of the structure until there is no significant reduction in the I/O rate.

REBUILDPERCENT: RACF supports REBUILDPERCENT for coupling facility cache structures. This support causes a rebuild to be driven for a RACF structure when the specified overall percentage of system-weight loses connectivity to the structure. Your installation should have an alternate coupling facility available for this purpose; if your installation only has one coupling facility, you should not use REBUILDPERCENT.

System weights are specified in the sysplex failure management policy. Note that different systems can have different system weights. REBUILDPERCENT is specified with the structure in the coupling facility resource management policy. We recommend that you use the same REBUILDPERCENT value for all RACF structures. What that percentage should be depends on the weights given to the systems and on your installation’s particular needs for connectivity. See *z/OS MVS Setting Up a Sysplex* for more information.

Note: RACF’s support for REBUILDPERCENT must be installed on each system in the sysplex. Systems running lower levels of RACF than the OS/390 Release 3 Security Server must have the appropriate PTF installed:
- UW90292 for FMID HRF2210
- UW90293 for FMID HRF2220

Reconfiguring RACF structures: Depending on the performance requirements of other structures in the coupling facility, you might want to change the coupling facility resource management policy in order to change the size of RACF structures or relocate them to another coupling facility. A rebuild is necessary to implement policy changes. RACF supports the rebuild interface to do this. See *z/OS MVS Programming: Sysplex Services Guide* for information on the rebuild interface. See “RACF Support of the Rebuild Interface” on page 314 for information on RACF’s support of the rebuild interface.

System authorization facility (SAF)

The system authorization facility (SAF) provides a system that gets control in response to a request from a resource manager. SAF conditionally directs control to RACF, if RACF or an installation-supplied processing routine, or both, is present. SAF does not require any other licensed program as a prerequisite, but overall system security functions are greatly enhanced and complemented by the concurrent use of RACF. The key element in SAF is the SAF router (ICHSFR00).

The SAF router

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. The resource-managing components and subsystems call the SAF router as part of certain decision-making functions in their processing, such as access control checking and authorization-related checking. These functions are

called *control points*. This single SAF interface encourages the use of common control functions shared across products and across systems.

The SAF router is always present on an MVS system whether or not RACF is present. If RACF is available in the system, the SAF router might pass control to the RACF router (ICHRFR00). The RACF router in turn invokes the appropriate RACF function, based on parameter information and the RACF router table. The RACF router table, consisting of modules ICHRFR0X and ICHRFR01, associates router invocations with RACF functions. If your installation decides not to call RACF, you must code the SAF router exits appropriately. For more information, see *z/OS Security Server RACROUTE Macro Reference*.

The SAF callable services router

For RACF callable services, the SAF callable services router performs a function similar to that of the SAF router. The SAF callable services router installation exit IRRSXT00 can be used to add to or replace the functions provided by RACF's callable services for z/OS UNIX System Services.

IRRSXT00 is called each time a RACF callable service for z/OS UNIX System Services is invoked. The exit is called both before and after RACF is called. When IRRSXT00 is called before RACF, the exit can request that RACF not be called. For more information, refer to *z/OS Security Server RACF Callable Services*.

Associating started procedures and jobs with user IDs

A procedure (PROC) consists of a set of job control language statements that are frequently used together to achieve a certain result. PROCs usually reside in the system procedure library, SYS1.PROCLIB, which is a partitioned data set. A started procedure is normally started by an operator, but can be associated with a functional subsystem. For example, DFSMS is treated as a started task even though it does not need to be specifically started with a START command.

Only RACF-defined users and groups can be specifically authorized to access RACF-protected resources. However, started procedures have system-generated JOB statements that do not contain the USER, GROUP, or PASSWORD parameter.

To enable started procedures to access the same RACF-protected resources that users and groups access, started procedures must have RACF user and group identities. By assigning them RACF identities, your installation can give started procedures specific authorization to access RACF-protected resources. For example, you can allow JES to access spool data sets.

As with any other user ID and group name, the user ID and group name that you assign to a started procedure must be defined to RACF using the ADDUSER and ADDGROUP commands. We recommend that you define the user ID to be a protected user ID, so that the user ID cannot be revoked by incorrect password attempts or used to enter the system in ways that require a password.To define a user ID as protected, assign it the NOPASSWORD and NOOIDCARD attributes using the ADDUSER or ALTUSER command. You might also need to use the PERMIT command to authorize the users or groups to get access to the required resources. For descriptions of the commands, see *z/OS Security Server RACF Command Language Reference*. For information on protected user IDs, see*z/OS Security Server RACF Security Administrator's Guide*.

The started procedure name is always available to the exit routines, whether or not the name is coded in the module. It is available in the parameter list for RACROUTE REQUEST=VERIFY exits and in the ACEE for RACROUTE REQUEST=AUTH and RACROUTE REQUEST=DEFINE exits.

If a started procedure is executed without associating its name with a RACF-defined user ID and group name, the started procedure runs as an undefined user. The procedure can access RACF-protected resources if the universal access authority for the resource is sufficient to allow the requested operation. However, if a started procedure uses a RACF-protected resource that grants or denies authority based on access list entries, you *must* associate the started procedure with a RACF-defined user ID and group name.

No user verification (password checking) takes place for a started procedure's user ID. However, you should still specify a password on the ADDUSER command for a started procedure. If you do not specify a password, RACF uses the user ID default group as the password. Any user who knows the started procedure's default group can use the user ID and default password to access the system.

RACF allows a started task or job to run even if the user ID is revoked.

RACF allows you to specify that a started procedure is *privileged*; this means that most RACROUTE REQUEST=AUTHs done for the procedure are considered successful, without actually performing any checking. In these cases, RACF:

- Does not call any exit routines
- Does *not* generate any SMF records
- Does not update any statistics

This bypassing also applies to the checking done for the CHKAUTH operand on the RACROUTE REQUEST=DEFINE macro instruction. All other RACF processing occurs as usual.

RACF allows you to specify that a started procedure is *trusted*; this means that most RACROUTE REQUEST=AUTHs done for the procedure are considered successful, without actually performing any checking. In these cases, RACF:

- Does not call any exit routines
- *Does* generate SMF records based on the audit options specified in SETROPTS LOGOPTIONS and the UAUDIT setting in the user ID profile
- Does not update any statistics

This bypassing also applies to the checking done for the CHKAUTH operand on the RACROUTE REQUEST=DEFINE macro instruction. All other RACF processing occurs as usual.

The trusted bit is used in a B1 system to indicate that the entry is part of the trusted computing base.

The TRUSTED attribute can be assigned to started procedures that have the ability to prevent a successful IPL or that need to create or access a wide variety of unpredictably named data sets. Candidates for the TRUSTED attribute include the following started procedures: JES, LLA, CATALOG, DUMPSRV, IEEVMPCR, SMF, VLF, VTAM, APSWPROC, RACF (if RRSF is used), IXGLOGR (if sysplex communications is used), and XCFAS (if sysplex communications is used.)

A trusted or privileged started task is treated as a z/OS UNIX System Services superuser if *any* z/OS UNIX user identifier (UID) is assigned to it in the OMVS segment. It does not have to have a UID of 0 to be considered a superuser.

Notes:

1. If ENTITY=(...,CSA) or ENTITY=(...,PRIVATE) is coded on the RACROUTE macro instruction, RACF ignores the privileged and trusted attributes and performs normal authorization processing.
2. Except as mentioned in Note 1, a started procedure that has the privileged attribute accepts any checking done by RACROUTE REQUEST=AUTH, including category and security-level checking, and returns a return code of 0 (access allowed). A started procedure can also access resources during failsoft processing without having RACF prompt the operator for permission. (For a discussion of failsoft processing, see “Failsoft processing” on page 102.)
3. While the trusted and privileged attributes are usually associated with started tasks, a RACROUTE REQUEST=VERIFY exit can mark other ACEEs privileged or trusted. RACF then processes those users in the same way as it does trusted or privileged started tasks.

Methods for associating started procedures with RACF identities

RACF provides two ways to assign RACF identities to started procedures:

- The started procedures table (ICHRIN03)
- The STARTED class

To modify the security definitions for started procedures using the started procedures table, you must edit the table, assemble and link-edit the updated table, and then re-IPL the system. The STARTED class allows you to modify the security definitions for started procedures dynamically, using the RDEFINE and RALTER commands, with no need to modify code or re-IPL. The STARTED class also allows you to process job names (on MVS/ESA 5.1) in addition to started procedure names.

When RACROUTE REQUEST=VERIFY(X) is issued with a started procedure name, RACF checks whether the STARTED class is active. If it is active, RACF uses the STARTED class to determine the user ID, group name, trusted flag, and privileged flag to use. If the STARTED class is not active, RACF uses the started procedures table (ICHRIN03). RACF also uses the started procedures table, and issues message IRR813I or IRR814I if the STARTED class is active but one of the following occurs:

- RACF cannot find a matching profile in the STARTED class.
- RACF finds a matching profile but the profile does not assign a user ID.

You must have a started procedures table (ICHRIN03) even if your installation uses the STARTED class. RACF cannot be initialized if ICHRIN03 is not present. A dummy ICHRIN03 is shipped with and installed by RACF. If you have replaced the dummy ICHRIN03 with your own version and want to delete your version, you must provide a dummy version with a halfword count field of X'0000' or X'8000'. We recommend that you leave your existing ICHRIN03 in place if you choose to use the STARTED class, in case, for example, someone unintentionally deactivates the STARTED class.

For installations that have an existing started procedures table and want to use the STARTED class, a sample REXX exec is provided in member ICHSPTCV in SYS1.SAMPLIB to process the output of ICHDSM00 and build RDEFINE commands to duplicate an existing started procedures table.

The STARTED class

The STARTED class allows you to assign RACF identities to started procedures and jobs dynamically, using the RDEFINE and RALTER commands. Unlike the started procedures table, it does not require you to modify code or re-IPL in order to add or modify RACF identities for started procedures. It provides, in effect, a *dynamic started procedures table*.

The MVS START command can start jobs as well as procedures. The START command specifies the member name to start and the job name to use. The member name is the name of a member of a partitioned data set that contains the source JCL for the task or job to be started. Using the STARTED class, RACF can assign different user IDs and group names to the same started member, depending on the job name that is used. CICS can use this, for example, to allow one procedure to be used for a variety of different CICS regions, which might have different security requirements.

Resource names in the STARTED class are of the form *membername.jobname*; for example, `CICS.JOBA`, `CICS.REGION2`, or `IMS.PROD`. The resource name is of the form *membername.membername* if no jobname is provided.

Profiles in the STARTED class have a segment, STDATA, containing fields for user ID, group name, trusted flag, privileged flag, and a trace flag. The user ID can be a RACF user ID or the character string `=MEMBER`, which indicates that the member name is to be used as the user ID. The group name can be a RACF group nameor the character string `=MEMBER`, which indicates that the member name is to be used as the group name. If tracing is specified, RACF issues operator message IRR812I during RACROUTE REQUEST=VERIFY or VERIFYX to indicate which profile is used. This message can be used during diagnosis of security problems with started procedures, to determine which profile was used for a particular started procedure.

The RDEFINE, RALTER, and RLIST commands define and modify profiles in the STARTED class. For more information on these commands, see *z/OS Security Server RACF Command Language Reference*.

We recommend that you define an appropriate generic profile that matches all possible START commands and that you specify either a user ID of limited privileges or `=MEMBER`. This approach ensures that, for any START command, there is always a matching profile with an STDATA segment that assigns a user ID. In addition, using this approach avoids the following situations, which cause RACF to use ICHRIN03 to process the START command:

- There is no matching profile.
- There is a matching profile, but it does not have an STDATA segment.
- There is a matching profile with an STDATA segment, but no user ID is specified.

Note: When the STARTED class is active, RACF uses it before using the started procedures table, ICHRIN03. It will override all the entries in ICHRIN03.

For additional information on the STARTED class, see *z/OS Security Server RACF Security Administrator's Guide*. For information on jobs and started tasks, see *z/OS MVS System Commands*.

The started procedures table (ICHRIN03)

The started procedures table (ICHRIN03) provides a way for your installation to assign RACF identities to your started procedures. It does not allow you to assign RACF identities to jobs; for that you must use the STARTED class.

RACF allows the started procedures table to contain a generic entry, indicated by an asterisk (*) in the procedure-name field. When searching the table for a procedure-name match, if RACF finds a procedure name of "*" as the last entry in the table and the procedure name was not specifically matched by any other entry in the table, RACF uses the "*" entry as a match for the procedure name. See also "Generic entry in ICHRIN03" on page 99.

Coding the started procedures module

To enable you to give RACF identities to started procedures, RACF provides the ICHRIN03 module. There are no entries in the module when you receive it from IBM. To use the started procedures table, you replace that module with your own table that associates the names of started procedures with user IDs and group names.

The table becomes part of the link pack area. After replacing the module, you must re-IPL the system with the CLPA option for the new module to be in effect. (You could also load the module into the MLPA, so that the link pack area does not have to be re-created.) You can specify either RMODE(24) or RMODE(ANY) for ICHRIN03.

The module (ICHRIN03) must consist of a table in the following format. See the RACTABLE member in SYS1.SAMPLIB for a sample started procedures table.

- **Number of entries:** A halfword of binary data containing a count of the entries in the table. If the high-order (leftmost) bit is turned on, this indicates that the table consists of 32-byte entries, the format used in RACF Version 1 Release 6 or later. If the high-order (leftmost) bit is off, this indicates that the table consists of 24-byte entries. (Use X'0000' or X'8000' if there are no entries.)
- **An array:** Each entry consists of 32 bytes of data. The first 24 bytes of character data show the started procedure name and its associated user ID and group name. Format each entry as follows:
  - Started procedure name: 8 bytes of character data. The name is required. The started procedure name must be left-justified and padded on the right with blanks.
  - User ID: 8 bytes of character data. A user ID is required. The user ID (or an equal sign for the generic entry) must be left-justified and padded on the right with blanks. (The maximum length of a user ID is 8 characters.)

    The user ID specified must be a RACF-defined user ID or an equal sign (=). The equal sign is valid only on the generic entry. See "Generic entry in ICHRIN03" on page 99.
  - Group name: 8 bytes of character data. The group name is optional.

    If a group name (or an equal sign for the generic entry) is used, it must be left-justified and padded on the right with blanks. If a group name is not used, this field must contain blanks.

    If the group name is specified, the user ID must be connected to this group. If a group name is not specified, the user ID's default group is used to build the ACEE used to grant authority to the started procedure.

– Flags: 1 byte of binary data. Setting bit 0 on (X'80') indicates that this entry has the privileged attribute. Setting bit 1 on (X'40') indicates that this entry has the trusted attribute.

  If both bits are on, the privileged attribute overrides the trusted attribute and no auditing is done.

  Even if a trusted or privileged attribute is specified, an equal sign or a RACF-defined user ID must be specified in the user ID field of the entry. For an equal sign, the started procedure name must also be a RACF-defined user ID.

  The remaining 6 bits must be zeros. (See notes.)
– Reserved: 7 bytes of binary data. These 7 bytes must be binary zeros.

Note: If you add a started procedure to the table, be sure that you increment the count field at the beginning of the table. Or, code your started procedures table so that the assembler calculates the count at assembly time, as shown in Figure 9. In this example, the high-order bit in the count field is set on to indicate that these are 32-byte entries. Adding 32 768, the decimal equivalent of X'8000', turns on the high-order bit.

```
ICHRIN03   CSECT
COUNT      DC     AL2(((ENDRIN03-COUNT-2)/32)+32768)
*--------- First Entry -----------------------------------------
ENTRY1     EQU    *
PROC1      DC     CL8'PROC1   '
USERID1    DC     CL8'TSO1    '
GROUP1     DC     CL8'SYS1    '
FLAGS1     DC     XL1'00'
           DC     XL7'00'
*--------- Last Entry ------------------------------------------
ENTRY2     EQU    *
PROC2      DC     CL8'*       '
USERID2    DC     CL8'TSO2    '
GROUP2     DC     CL8'=       '
FLAGS2     DC     XL1'00'
           DC     XL7'00'
*---------------------------------------------------------------
ENDRIN03   EQU    *
           END
```

Figure 9. Coding ICHRIN03 So the Assembler Calculates the Count Field

Generic entry in ICHRIN03

The started procedures table can contain one generic entry, indicated by an asterisk (*) in the procedure-name field. The generic entry enables you to add started procedures to your system without requiring an IPL to update ICHRIN03. For this reason, we recommend that you have a generic entry.

The generic entry must be the last entry in the table; otherwise, it is ignored. The corresponding user ID in this entry can be a valid user ID or an equal sign (=). The group name specified in the table entry can be either blanks, a valid group name, or an equal sign (=).

Note: You can use the equal sign only for a generic started procedures table entry; it is not valid for non-generic entries.

When searching the table for a procedure-name match, if RACF finds a procedure name of asterisk (*) as the last entry in the table and the procedure name was not specifically matched by any other entry in the table, RACF uses the asterisk (*) entry as a match for the procedure name.

If a user ID is specified for the asterisk (*) entry, RACF associates that user ID with the started procedure name. If the user ID field contains an equal sign (=), RACF uses the procedure name that was matched with the generic entry asterisk (*) as the user ID.

If the group name is blank, the started procedure will run using the default group in the profile record for the specified user ID (specified on the ADDUSER command). If the group-name field contains an equal sign (=), RACF uses the procedure name that was matched with the generic entry asterisk (*) as the group name.

If the generic entry has an equal sign (=) for the user ID (or group name), the procedure name that matches the equal sign must be defined to RACF as a user ID (or group name); otherwise the procedure runs as an undefined RACF user (*user ID* = *).

The user ID and the group name cannot both contain values of equal sign (=) in the asterisk (*) procedure-name entry of the table because it is not possible to have a RACF user and group with the same name. During RACF initialization, RACF inspects the table entries for a possible generic entry. If RACF finds a generic entry, and it is not the last entry, or if it contains an equal sign (=) in both the user ID and group name fields, the system issues message ICH522I. This condition does not prevent RACF from being initialized. During execution, RACF ignores all the entries that are not valid, and all procedures that do not have an exact match in the table run as undefined users.

If you do not specify an asterisk (*) in the table, RACF uses the RACF default user ID asterisk (*) and group name asterisk (*) for authorization checking.

The started procedures table (ICHRIN03) can include an entry indicated by an asterisk (*) in the procedure name field as the last entry in the table. The following examples show the possible formats of the asterisk (*) procedure-name entry. Note that none of these examples has the privileged flag bit on.

Attention

Do not specify your generic entry with equal sign (=) in the user ID field and blanks in the group-name field because this can allow a procedure to run illegally with the identity of a valid user ID.

Avoid this problem by following this scenario:

1. Create a valid RACF group, for example, PROCGRP.
2. Place the group name (PROCGRP) in the group field of the generic entry.
3. Connect all started-procedure user IDs—that *only* run as started procedures—to PROCGRP.

Be careful which libraries your started procedures come from and do not let your users update them. Refer to the JES customization manuals for information on specifying procedure libraries.

Example 1

COUNT	PROC.	USER ID	GROUP	FLAGS	RESERVED
X'8002'	PROC1	TSO1	SYS1	00000000	7 bytes of X'00'
	*	TSO2	=		

If RACF searched the started procedures table in Example 1 for the procedure name PROC2, it would not find a specific match. RACF would consider the asterisk (*) entry in the table as a match for procedure PROC2. The RACF user ID associated with PROC2 is TSO2, and the group name is PROC2.

If RACF searched the started procedures table in Example 1 for PROC1, it would find a specific match and associate PROC1 with user ID TSO1 and group SYS1.

Example 2

COUNT	PROC.	USER ID	GROUP	FLAGS	RESERVED
X'8002'	PROC1	TSO1	SYS1	00000000	7 bytes of X'00'
	*	=	PROCGRP		

If RACF searched the started procedures table in Example 2 for the procedure name PROC2, it would not find a specific match. RACF would consider the asterisk (*) entry in the table as a match for procedure PROC2. The RACF user ID associated with PROC2 is PROC2, and the group name is PROCGRP.

Note: In this example, PROC2 is a valid RACF user ID. It has been connected, by means of CONNECT, to PROCGRP, a valid RACF group. All other valid user IDs that are started procedures should also be connected to this group through CONNECT. Then, if a started procedure happens to have the same name as a valid RACF user ID, when RACF searches the started procedures table RACF does not find a match for the procedure, because the user ID is not connected to group PROCGRP. The procedure runs, but it runs as the default user ID, and does not have access to resources that the valid RACF user ID has access to.

Example 3

COUNT	PROC.	USER ID	GROUP	FLAGS	RESERVED
X'8001'	PROC1	TSO1	SYS1	00000000	7 bytes of X'00'

If RACF searched the started procedures table in Example 3 for the procedure name PROC2, it would not find a specific match. RACF would associate with PROC2 the default user ID asterisk (*) and the default group name asterisk (*).

If RACF searched the started procedures table in Example 3 for PROC1, it would find a specific match and associate PROC1 with user ID TSO1 and group SYS1.

Example 4 (Error in Table)

COUNT	PROC.	USER ID	GROUP	FLAGS	RESERVED
X'8002'	*	TSO2	SYS2	00000000	7 bytes of X'00'

COUNT	PROC.	USER ID	GROUP	FLAGS	RESERVED
	PROC1	TSO1	SYS1		

Because the started procedures table in Example 4 contains a generic entry but the generic entry is not the last entry in the table, RACF issues an error message during RACF initialization and ignores the generic entry whenever it searches the table. If RACF searched the started procedures table in Example 4 for the procedure name PROC2, it would not find a specific match. RACF would associate with PROC2 the default user ID asterisk (*) and the default group name asterisk (*).

If RACF searched the started procedures table in Example 4 for PROC1, it would find a specific match and associate PROC1 with user ID TSO1 and group SYS1.

Example 5

COUNT	PROC.	USER ID	GROUP	FLAGS	RESERVED
X'8002'	RACF	RACFAS		01000000	7 bytes of X'00'
	IRRDPTAB	IRRDPI00		00000000	7 bytes of X'00'

This is an example of entries you will need if you plan to activate the RACF subsystem and dynamic parse, and use the RACF remote sharing facility (RRSF). The entry for the RACF subsystem is marked trusted, to give the RACFAS user ID access to the resources used by RRSF functions.

Changing the ICHAUTAB module

The RACF authorized caller table contains the names of programs that your installation authorizes to issue RACROUTE REQUEST=LIST or RACROUTE REQUEST=VERIFY without the NEWPASS keyword. The programs must be reentrant and fetched from an APF-authorized library.

Incorrect use of ICHAUTAB causes system integrity problems; therefore, we recommend that you not use ICHAUTAB, but rather run these programs with APF-authorization. If you cannot make the programs APF-authorized, see the additional information in “Changing the ICHAUTAB Module” on page 363.

RACF sysplex communication: If you use ICHAUTAB when RACF is enabled for sysplex communication, the table should reside in a common library and be shared by all members of the data sharing group.

Failsoft processing

Failsoft processing occurs when no data sets in the primary RACF database are available (RACF is installed but inactive). Although it degrades system performance and system security, in rare cases it might be necessary when you repair RACF. During failsoft processing RACF cannot make decisions to grant or deny access. For data sets, RACF prompts the operator frequently to grant or deny access. For general resource classes, RACF returns a return code of 4 and the resource manager (for example TSO or CICS) decides on the action.

There are several reasons why failsoft processing might be in effect on your system:

- RACF is installed but does not know the name of the master primary data set.
- Failures occurred during RACF initialization at IPL time.
- An RVARY INACTIVE command was issued (inactivating all data sets in the primary database).

When RACF is enabled for sysplex communication, failsoft processing also results when:

- The system is in XCF local mode.
- The RACF database does not reside on a shared device.
- A system attempting to join an existing RACF data sharing group is unable to allocate one or more data sets defined by the first system. The system operator is not prompted for a database name, and the system joins the group and begins failsoft processing.
- RACF encounters an internal error while processing a request on behalf of the RACF data sharing group.

When RACF is in data sharing mode, failsoft processing also results when the system is running an MVS release that does not support the RACF sysplex data sharing option.

Failsoft can be temporary or permanent. *Temporary failsoft* occurs as a result of the RVARY INACTIVE command. You can exit temporary failsoft by issuing the RVARY ACTIVE command. *Permanent failsoft* occurs as a result of a serious sytem error. You must re-IPL the system to exit permanent failsoft.

The logging your installation specified while RACF was active remains operative after failsoft processing goes into effect. In addition, RACF logs all accesses that the operator allows or denies.

RACF calls the RACROUTE REQUEST=AUTH and RACROUTE REQUEST=DEFINE preprocessing exit routines during failsoft processing. The use of preprocessing RACF exits enables an installation to define its own version of failsoft processing so that it can avoid the system performance problems caused by continual operator prompts. For example, an exit could be written to record resource definitions in SMF records and later automatically apply them to the RACF database.

General considerations

The following considerations apply when the RACF database is inactive (failsoft processing occurs):

- If RACF enters failsoft during initialization, you must re-IPL.
- If RACF *is not* enabled for sysplex communication, a RACF database that is shared by two systems is deactivated only for the system from which you enter the RVARY command. You should deactivate a database from all systems that share it, or results might be unpredictable.

  When RACF *is* enabled for sysplex communication, certain RVARY commands (SWITCH, ACTIVE, INACTIVE, DATASHARE, NODATASHARE) are propagated from the system on which the command is entered to each of the other RACF members of the data sharing group.
- If failsoft processing is in effect, whenever a user attempts to access a data set RACF sends a message to the operator to request access. The operator then decides whether to allow access to that data set and sends a response to RACF.

Before you deactivate the RACF database, ensure that the operator is prepared for the large number of prompts that will result.

- The operator's ability to allow user access to data sets when failsoft processing is in effect will probably not be sufficient to keep the system running error-free. You might experience failures in many system functions, such as TSO user logons, CICS user signons, and batch jobs and started tasks that need data contained in the database.
- The RACROUTE REQUEST=AUTH and RACROUTE REQUEST=DEFINE postprocessing exit routines do not gain control when RACF failsoft processing is active.
- Attempts to define resources to RACF with RACROUTE REQUEST=DEFINE processing cause an operator information message. The DEFINE request terminates with a return code of zero. After RACF is reactivated, examine the information in the operator messages and use the ADDSD or RDEFINE command or both to define appropriate profiles.

The following considerations apply when a subset of the data sets in the RACF database are inactive (failsoft processing does not occur):

- Batch and TSO users whose profiles are on a deactivated data set can enter the system as if RACF were not installed, assuming the TSO users have entries in the SYS1.UADS data set.
- You cannot enter RACF commands to make changes to profiles on a deactivated data set.
- If you have more than one data set in your primary database, you must enter RVARY INACTIVE for all of your primary data sets for failsoft processing to be in effect. If you enter RVARY INACTIVE for only one of the primary data sets, failsoft processing will not be in effect; therefore, any RACF activities involving that data set will fail.
- You can use exit routines to examine the data set descriptor table created during RACF initialization and determine if a data set in the RACF database has been deactivated by the RVARY command.

Impact on users

Failsoft processing affects you in the following ways:

- If you are already logged on:

  If RACF is in failsoft mode, those users already on the system continue to have certain access requests validated by RACF. These requests are data-set-related requests or RACROUTE REQUEST=FASTAUTH processing. To continue validating, RACF uses whatever in-memory tables are still valid in addition to routing control to various exits for further processing. RACF can also continue to log access requests, whether it grants them or not.

  Note: If the user requests access to a data set and the decision could not be made using a valid internal table, RACF, through failsoft processing, prompts the operator to approve the request.

- If you are not logged on:

  The only users who can log on to TSO are those who have user IDs in SYS1.UADS and know their UADS password. These users are not known to RACF and RACF prompts the operator each time one of them requests access to a general resource or a data set that does not start with the user's ID. (This occurs because the users had not been verified—no password checking was done by RACROUTE REQUEST=VERIFY.)

If you reactivate RACF, you should have the users log off and log back on so that they can be identified to RACF.

CICS considerations

RACF allows CICS to establish defaults for CICS information such as Operator Identification, Operator class, Operator Priority, XRF Re-signon option, Terminal Timeout Value, and User Data Area.

For further information, see *CICS RACF Security Guide*.

CICS timeout value

RACF allows you to specify CICS timeout values in the form:

- M (single-digit minutes; for example, 5)
- MM (double-digit minutes; for example, 20)
- HMM (single-digit hours, double-digit minutes; for example, 234 for 2 hours and 34 minutes)
- HHMM (double-digit hours, double-digit minutes; for example, 1230 for 12 hours and 30 minutes)

You can configure your system to use either the limited range of values that CICS releases prior to 4.1 support (00 to 60 minutes) or the expanded range of values. To do this, edit the RACF-supplied panel ICHPKEYS (or the ICHPKYUM SYS1.SAMPLIB member as a model) and modify the keyword CTIMEOUT. Change CTIMEOUT to `HHMM` to use the expanded range of values, or to `MM` to use the limited range of values. Apply the modified ICHPKEYS using an SMP/E USERMOD so that it will not be lost if maintenance is applied.

The default value of CTIMEOUT when RACF is installed is `HHMM`. If you are running a release of CICS previous to 4.1, you should change CTIMEOUT to `MM`.

Note: If you change the default panel configuration using SMP/E and need to install a PTF that affects ICHPKEYS, the PTF will not install on the first pass. You must run an additional APPLY step telling SMP/E to apply the PTF on the USERMOD to ICHPKEYS and refit the modification. RACF minimizes the rework by shipping the sample USERMOD in SYS1.SAMPLIB whenever a new copy of ICHPKEYS is shipped.

DFP considerations

RACF allows the Data Facilities Product and DFSMS/MVS® 1.1.0 or later to establish defaults for the DFP constructs known as storage class, management class, data class, and data application on the RACF database. These constructs are stored by field name in the DFP segment of the USER and GROUP profiles.

The DFP field, RESOWNER, contains the user ID or group name of the owner of the data set, rather than the owner of the profile. In general, the data set profile contains a specified RESOWNER field when the data set resource owner differs from the data set profile's high-level (first) qualifier.

Using a combination of the FIELD class and the command processors, the RACF administrator can decide which fields users can define and update in their DFP segment.

You should issue a SETROPTS RACLIST command with the STORCLAS and MGMTCLAS classes to improve performance.

For further information, see *z/OS Security Server RACF Command Language Reference* and *z/OS Security Server RACF Security Administrator's Guide*.

TSO considerations

The RACF database includes a TSO segment where TSO can store TSO user logon information. Thus, TSO has an alternative to storing TSO user information in the UADS data set.

Using a combination of the FIELD class and the command processors, the RACF administrator can decide which fields users can define and update in their TSO segment.

Attention: You should keep at least one user definition in the UADS data set for emergency use. This is further discussed in "Sysplex recovery scenarios that require XCF-local mode" on page 314.

For further information, see *z/OS Security Server RACF Command Language Reference* and *z/OS Security Server RACF Security Administrator's Guide*.

ISPF considerations

Depending on how you have customized ISPF, when a user issues a command from an ISPF environment ISPF might write the TSO command buffer to the ISPLOG data set. You should consider preventing this action for RACF commands that can contain sensitive information, such as ALTUSER, PASSWORD, RACLINK and SETROPTS. Note, however, that if you do this, *no* ALTUSER, PASSWORD, RACLINK, or SETROPTS commands will be written to the ISPLOG data set.

You can use the ISPF TSO command table (ISPTCM) to control processing for TSO commands. To prevent ISPF from writing the TSO command buffer to the ISPLOG data set for a particular command, you must add an entry to the ISPTCM for that command with the '...1....' bit set in the FLAG field. For information on customizing the ISPTCM, see *z/OS ISPF Planning and Customizing*.

DB2 considerations

You can use RACF to protect DB2 data by installing the RACF/DB2 external security module. The module is an exit load module that can be used as the DB2 access authorization exit routine. The module receives control from the DB2 access control authorization exit point and allows you to control the access to DB2 objects by defining profiles in the RACF database.

For more information, see Chapter 6, "The RACF/DB2 external security module," on page 187.

DASD data sets

This section describes:

- Using utilities on RACF-protected DASD data sets. The system utilities for which RACF performs authorization checking are listed, and some special rules that you must consider when using utilities on RACF-protected data sets are discussed.
- Moving a RACF-indicated DASD data set between systems. Possible situations and some considerations when moving RACF-indicated data sets from one system to another are presented.
- Using access method services commands. Using access method service commands with RACF-protected VSAM data sets is described.

Also see “DASD volumes” on page 113.

There are special considerations when you set up automatic direction of application updates for updates to discrete data set profiles. For information on these considerations, see *z/OS Security Server RACF Security Administrator's Guide*.

Using utilities on RACF-protected DASD data sets

RACF performs authorization checking for RACF-protected DASD data sets that are accessed by the following system utilities when the utilities issue the OPEN macro instruction:

ADRDSSU	IEBEDIT	IEBUPDTE
ICKDSF	IEBGENER	IEHATLAS
IEBCOMPR	IEBISAM	IEHLIST
IEBCOPY	IEBPTPCH	IEHMOVE
IEBDG		

To use these utilities on RACF-protected data sets, you must be defined to RACF and must be permitted access to any RACF-protected data sets that the utilities access (unless the UACC for the resource is sufficient to allow access).

Notes:

1. You can use standard or nonstandard naming conventions when you define DASD data set profiles to RACF. By default (the standard naming convention), RACF expects the high-level qualifier of the name of a data set profile to be either a RACF-defined user ID or group name.

   RACF also allows you to use options to modify existing data set names to make them conform to RACF standard naming conventions. For example, the single-level name prefixing facility of RACF adds a qualifier to make the data set name acceptable to RACF routines. If you are not familiar with the options for nonstandard naming conventions, see *z/OS Security Server RACF Security Administrator's Guide* for more information.

   You also have the ability to create a naming convention table (ICHNCV00), which RACF uses to check the data set name in all commands and SVCs that process data-set names. Creating this table will help you set up and enforce data-set naming conventions that are different from the standard RACF naming convention. For more information, see “Data set naming convention table” on page 251 and the description of the ICHNCONV macro in *z/OS Security Server RACF Macros and Interfaces*.
2. See Chapter 7, “RACF database utilities,” on page 201 for a description of the RACF utilities that can be used on the RACF database.

The following topics describe special rules you must consider when using utilities on RACF-protected data sets.

Using utilities with the OPERATIONS or group-OPERATIONS attribute

A user who has the OPERATIONS attribute can:

- Create, rename, and define group data sets for groups except when both of the following are true:
  - The user is connected to the group with less than CREATE authority
  - The user has less than ALTER access to the data set if it is protected by a generic profile
- Create, rename, and define user data sets that are prefixed by another user's user ID (unless, for rename, specifically prohibited in the data set's access list).

Thus, a user with the OPERATIONS attribute can perform many operations on DASD data sets using the system utilities.

The group-OPERATIONS user can perform all operations that can be performed by the OPERATIONS user; however, the authority is limited to the resources that are within the scope of the group to which the user is connected with the group-OPERATIONS attribute. For more information on the scope of the group, see *z/OS Security Server RACF Command Language Reference*.

Renaming RACF-protected data sets

You can rename a DASD data set that is protected by a discrete or generic RACF profile using the IEHPROGM utility, the access method services ALTER command, or the TSO RENAME command. IEHMOVE can also rename a data set but does so by creating a new data set having the new name. The following rules apply when renaming a data set:

- You cannot rename a multivolume, non-VSAM data set for which a discrete profile exists.
- You must have the OPERATIONS attribute (or group-OPERATIONS with the restrictions it carries) or have ALTER access authority to the data set.

  Note: If the data set is protected by a discrete profile, you cannot rename the data set to a name whose high-level qualifier is a group that you are connected to with less than CREATE authority, regardless of your OPERATIONS or group-OPERATIONS attribute.
- You must have the same authority to the new name as would be required to create it.
- The new name must conform to the RACF data set naming conventions, unless the naming convention table modifies the processing of data set names.
- If the data set is covered by a generic profile, you cannot rename it unless the new name is also covered by a generic profile and you have either ALTER authority to both new and old generic profiles or the OPERATIONS attribute (or group-OPERATIONS attribute with the restrictions it carries).
- You cannot rename an individual data set of a GDG if:
  - It is protected by a profile for the base portion of the GDG name.
  - The new name is a non-GDG name or is a GDG name for which there is no base profile defined.

To effectively rename a data set that cannot be renamed using IEHPROGM or TSO RENAME because of the above restrictions, copy the data set (using IEHMOVE) to one having the new name.

When you rename a data set that is protected by a discrete profile, RACF makes the following changes to the profile:

- If you do not have the OPERATIONS attribute (or group-OPERATIONS with the restrictions it carries) and the new name indicates a user data set (that is, the high-level qualifier is a user ID), the access list for the data set remains the same, but the profile is changed to show you as the owner.

  If you have the OPERATIONS attribute (or group-OPERATIONS with the restrictions it carries), the user whose user ID is the high-level qualifier of the renamed data set becomes the owner.

  In both cases, the profile changes to show the current connect group as the one under which the data set was renamed.
- If you have the GRPACC attribute, and the high-level qualifier of the old data set name is a group name, RACF removes the group name from the access list.

  Note: If the high-level qualifier of the new data set name is also a group name, RACF adds that group name to the access list. This action occurs even if the same group was removed in this step.
- If the new name indicates a group data set (the high-level qualifier is a group name), RACF updates the access list in the following way. Your user ID is added to the list and given ALTER authority, unless your user ID is already in the list. In this case, your authority remains unchanged.

  If you have the GRPACC attribute, the group indicated by the new name is added to the list and given UPDATE authority. The profile is also updated to show you as the owner of the data set (unless your authority to rename the data set is through your OPERATIONS or group-OPERATIONS attribute, in which case the owner is not changed) and to show the current connect group as the one under which the data set was renamed.

Attention

No change occurs in generic profiles applying to a data set being renamed. As a result of being renamed, a data set might be protected by a different generic profile from the one applied to the old name.

Using IEHMOVE with the ADSP attribute

The following rules apply when you use the IEHMOVE system utility with RACF-indicated DASD data sets and you have the ADSP attribute:

- Moved and copied data sets that follow the RACF naming convention for data sets are automatically defined to RACF for protection. (IEHMOVE fails if the new data set name does not follow the RACF naming convention.)
- You cannot create a data set whose name has a high-level qualifier that is not your own user ID (unless you have the OPERATIONS or group-OPERATIONS attribute or the data set being moved or copied is a group data set and you are connected to that group with at least CREATE access authority).
- You cannot move a data set with a target volume specified that is the same as the originating volume unless you code RENAME on the MOVE statement. If you do not code RENAME, IEHMOVE fails when it attempts to allocate an IEHMOVE-generated name that does not follow RACF naming conventions.

- If you select the option that prevents data sets with the same names from being defined to RACF with discrete profiles (by modifying the ICHSECOP module), then IEHMOVE cannot move or copy a data set that is RACF-protected with a discrete profile unless the new data set has a different name from the old data set.

Using IEHMOVE with the COPYAUTH parameter

On the MOVE and COPY statements of the IEHMOVE system utility, you can specify the COPYAUTH parameter. (Note: You cannot move a data set with a target volume specified that is the same as the originating volume unless you code RENAME on the MOVE statement.) The COPYAUTH parameter enables you to use the discrete profile of the old RACF-protected data set as a model to build a discrete profile for and RACF-indicate the new data set.

This modeling capability causes RACF to copy directly the following fields:

- Access lists
- Level
- UACC
- Warning and logging options (auditing flags)
- Installation data
- Security categories and security levels
- Erase option indicator
- User to be notified

The owner (the content of the owner field) is determined by the following rules:

- If the current user does not have the OPERATIONS or group-OPERATIONS attribute, then the user becomes the owner of the data set profile.
- If the current user has the OPERATIONS or group-OPERATIONS attribute, then either:
  - For a new user data set that has a different high-level qualifier from the modeled data set name, the user whose user ID is the high-level qualifier of the new data set name becomes the owner.
  - In all other cases, IEHMOVE copies the owner field directly from the model.

Note: A data set that is not RACF-indicated will not be protected after moving unless there is a suitable generic profile on the destination system.

Using the DFSMSdss and DSF utilities

For information on using the DFSMSdss and DSF utilities, see:

- *z/OS DFSMS Introduction*
- *z/OS DFSMSdss Storage Administration Guide*
- *z/OS DFSMSdss Storage Administration Reference*
- *Device Support Facilities User's Guide and Reference*

Moving a RACF-indicated DASD data set between systems

You can move a RACF-indicated DASD data set from system to system. Four situations are possible. You can move the data set from a system with RACF active to:

- Another RACF-active system (an MVS system with RACF installed and active, and with a RACF database different from the one used by the source system)

  Note: If the source and destination systems share the same RACF database, no special action is needed to protect the data set.

- A RACF-inactive system (an MVS system with RACF installed but with the bypass RACROUTE REQUEST=VERIFY option in effect or with RACF deactivated by the RVARY command)
- A non-RACF system with RACF indicator checking
- A non-RACF system

  Note: In this situation, the data set is not protected in any way on the non-RACF system.

Moving a RACF-indicated data set to a RACF-active system

When a RACF-indicated data set is moved to a system with RACF active, the data set might or might not be defined to RACF on the destination system. If the data set is not defined to RACF, you must define it on the destination system and enter the ADDSD command with the NOSET operand. You specify NOSET because the data set is already RACF-indicated. If the data set is already defined to RACF on the destination system, no additional steps are needed; the data set is fully RACF-protected.

Attention

The access lists should be identical; otherwise, a security exposure can exist.

You can move a RACF-indicated data set to a system that already has a RACF-defined data set with the same name. If the data sets reside on volumes with different serial numbers, enter the ADDSD command with the NOSET operand to define the data set separately. If they reside on volumes with the same serial number, the data sets share the same discrete profile. There is only one access list and one set of statistics and logging options.

Regardless of whether the data set is RACF-indicated, if its name matches a generic profile at a destination system that has RACF-active and generic profile checking enabled, the data set will automatically be protected if always-call is in effect. The generic profile that matches at the destination system can have attributes (such as an access list) totally different from the discrete or generic profile that applied to the data set at the source system.

Moving a data set with a discrete profile to a RACF-inactive system

When a RACF-indicated data set that is protected by a discrete profile is moved to a system with RACF inactive, attempts to access the data set on the destination system will cause RACF to ask the operator to allow access to a resource.

If you want access to a resource without the operator intervening, you can enter the DELDSD command on the source system (before moving the data set) to turn off the data set's RACF indicator and to delete the data set's RACF profile. On MVS, RACF protection for the data set no longer exists.

Moving a RACF-indicated data set to a non-RACF system with RACF indicator checking

When a RACF-indicated DASD data set is moved to a non-RACF system, attempts to access the data set on the destination system fail. To prevent the failure (abend), take one of the following actions:

- For either a VSAM or a non-VSAM data set, enter the DELDSD command on the source system (before moving the data set) to turn off the data set's RACF indicator and to delete the data set's profile.

- To turn off the RACF indicator for a non-VSAM data set but to preserve the profile, perform the following steps on the source system before the move:
  1. Enter the ALTDSD command with the ADDVOL and NOSET operands to add a dummy volume number to the data set's RACF profile.
  2. Enter the ALTDSD command with the DELVOL and SET operands to delete the volume number of the data set from the RACF profile and to turn off the RACF indicator. (The dummy volume number remains in the profile.)
  3. When you move the data set back to the source system, enter the ALTDSD command with the ADDVOL and SET operands to restore the volume number in the profile and to set the RACF indicator. Then enter ALTDSD with the DELVOL and NOSET operands to delete the dummy number.

Notes:

1. There is no way to turn off the RACF indicator for a VSAM data set and still preserve its discrete profile.
2. For both a VSAM data set and a non-VSAM data set, the data set will have RACF protection only if generic access checking is enabled and a generic profile applies; otherwise, it reverts to password protection if the data set is password protected.

Moving a multivolume RACF-indicated data set between systems

When moving a multivolume RACF-indicated DASD data set from a source system to a destination system, you might need to update the RACF database on the destination system. This step is necessary if you extend a non-VSAM data set to additional volumes on one system and then move it to another system where the data set is defined to RACF. In this case, the RACF database on the destination system does not have the new volume serial number in the data set profile. The procedure required to add the new volume to the profile depends on whether the data set was extended on a RACF-active system or a non-RACF (or RACF-inactive) system.

Note: This explanation assumes the data set is RACF-defined on the destination system. If it is not, you must enter the ADDSD command to define it.

If the data set was extended on a RACF-active system, you must enter the ALTDSD command (with the ADDVOL and NOSET operands) on the destination system. This command adds the new volume serial number to the RACF database profile but does not change the data set's RACF indicator.

Note: The source system automatically added the new volume number to its own RACF database profile when the data set was extended. At the same time, the source system set the RACF indicator for that portion of the data set that is on the new volume.

If the data set was extended on a non-RACF or RACF-inactive system, you must enter the ALTDSD command (using the ADDVOL and SET operands) on the destination system. In addition to adding the new volume serial number to the RACF database profile, this command sets the RACF indicator for that portion of the data set on the new volume. You use the SET operand because the non-RACF source system did not set the indicator when the data set was extended.

Multivolume VSAM data sets do not require these procedures. The RACF profile for a VSAM data set gives the volume serial number of only the catalog containing the data set entry. Therefore, a VSAM data set can extend to additional volumes without requiring changes to the RACF profile. Also, the VSAM catalog contains one

RACF indicator for the entire data set regardless of the number of volumes; an indicator does not need to be set for each new volume.

Using access method services commands

This section describes considerations when using the following access method services commands with RACF-protected VSAM data sets:

- LISTCAT
- REPRO
- RESETCAT
- IMPORT
- IMPORTRA

LISTCAT command

When you use the LISTCAT command on a RACF-protected VSAM data set and you have less than ALTER access authority to the data set, you might receive an authorization-failure message followed by listed information. It is likely that you requested a list of passwords, which requires ALTER access authority. VSAM writes the error message that indicates you do not have sufficient authority to list passwords and then lists the requested information (except passwords).

REPRO/RESETCAT/IMPORT/IMPORTRA commands

A discrete data set profile in the RACF database contains the volume serial number of the catalog for a RACF-protected VSAM data set. This volume serial number must match the volume serial number supplied on the RACROUTE REQUEST=AUTH macro instruction; otherwise RACF cannot locate the correct profile in the RACF database. (VSAM processing routines supply the volume serial number of the catalog on the RACROUTE REQUEST=AUTH macro instruction.)

The REPRO, RESETCAT, IMPORT, and IMPORTRA commands can cause the volume serial number of the catalog containing a RACF-protected data set to change and differ from the volume serial number in the data set's profile in the RACF database. These commands do not invoke RACF to update the profile in the RACF database. Therefore, the user who is maintaining the RACF-protected data set must update the data set profile with the correct volume serial number.

To update data set profiles, enter the ALTDSD command with the ALTVOL operand. Note that you can use the SEARCH command to build a command procedure containing ALTDSD commands for all VSAM data sets cataloged on the same volume.

Or, to update profiles, you can use the TSO command-procedure facility to:

1. Obtain a list of the catalog entries for a volume for RACF-indicated data sets by using the LISTCAT command.
2. Obtain a list of the RACF-indicated VSAM data sets by using the SEARCH command.
3. Compare the volume serial numbers of RACF-indicated data sets (obtained by the LISTCAT command) with the volume serial numbers of RACF-indicated data sets (obtained by the SEARCH command) to ensure the correctness and completeness of volume serial number information.

DASD volumes

This section presents considerations to be aware of when using the RACF DASDVOL authorization facility to authorize selected users and groups to DASD volumes that contain RACF-protected data sets.

Scratching DASD data sets

A user who has ALTER access authority to a DASD volume can scratch data sets on the volume whether or not the user is authorized access to the data sets. (If the user is not authorized to access the data set, RACF issues message ICH408I to the security console to report an access violation even though the data set is scratched.) When a data set is scratched, RACF deletes the discrete profile for the data set from the RACF database, or in the case of a multivolume data set, removes the volume serial number from the data set profile.

Moving DASD volumes between systems

When you move a DASD volume to a system that has RACF installed:

- If the DASD volume is defined on the new system, RACF performs authorization checking in the normal manner.
- If the DASD volume is not defined on the new system, or the DASDVOL class is not active, RACF protection or password protection is performed for individual data sets.

UCBs above 16MB

If you are running MVS/ESA 5.2 or higher, you can define unit control blocks (UCBs) above 16MB and:

- Place the RACF database on these devices.
- Use RACF to protect data sets on these devices.

See *z/OS HCD Planning* for more information.

Protecting tape data

This section provides information about RACF-protecting tape volumes and tape data sets. It discusses:

- Aspects of tape-data protection that a systems programmer might need to implement
- Using utilities on RACF-protected tape volumes
- Moving tape volumes and multivolume tape data sets from one system to another

There are special considerations when you set up automatic direction of application updates for updates to tape data set profiles. For information on these considerations, see *z/OS Security Server RACF Security Administrator's Guide*.

For more information on tape-volume protection, see *z/OS Security Server RACF Security Administrator's Guide*.

Tape data protection and bypass label processing (BLP)

You can use RACF to control the use of bypass label processing (BLP). If an installation specifies BLP at system generation or JES initialization, and the user specifies BLP on the LABEL parameter of the DD statement or on the dynamic allocation text unit, RACF issues a RACROUTE REQUEST=AUTH to the FACILITY class, resource ICHBLP, to determine if the user's BLP request can be honored. To activate this additional RACF protection, installations must do the following:

- Define the profile ICHBLP to RACF using the RDEFINE command
- Activate the TAPEVOL class (it is not necessary to define tape volumes to the TAPEVOL class).

An installation can add users or groups or both to the profile access list using the PERMIT command.

If BLP is specified on the LABEL parameter and the system does not support BLP, the tape is treated as a nonlabeled (NL) tape.

Considerations for unlabeled (NL) tapes

For a description of RACF considerations for opening nonlabeled tapes for input and output, see *z/OS Security Server RACF Security Administrator's Guide*.

You should be careful when using nonspecific volume requests for output volumes because the operating system assigns volume serial numbers and it is impossible for you to determine if the volume mounted is defined to RACF under a different number. If your installation plans to use RACF protection for nonlabeled tapes, you should use JCL scans to prevent the use of nonspecific volume requests and institute procedures to ensure that operators mount the correct tapes. In addition, the installation must activate the TAPEVOL class.

Using utilities on RACF-protected tape volumes and tape data sets

With the exception of IEHINITT, RACF performs authorization checking for tape volumes that are accessed by system utilities when these utilities issue the OPEN macro instruction; therefore, users of system utilities must be defined to RACF and have authorized access to any RACF-protected tape volumes that the utilities access.

The installation should restrict the use of the IEHINITT utility to only authorized administrators. You can use the RACF program control option to restrict the utility.

Moving tape volumes between systems

When a tape volume is moved to a system that has RACF installed and the tape volume protection option is active, then:

- If the tape volume is defined in the RACF database of the new system, RACF performs authorization checking in the normal manner.
- If the tape volume is not defined in the RACF database of the new system, RACF performs authorization checking and the result is "not defined." Password checking is then done.

If the tape volume protection option is not active on the new system, RACF does not perform authorization checking.

Moving multivolume tape data sets between systems

When moving a multivolume tape data set that resides on a RACF-protected tape volume set, you might need to update the profile for the tape volume set on the destination system. This step is necessary if the multivolume data set was extended to additional volumes on one system and is then moved to another system where the tape volume set is defined to RACF. To update the profile for the tape volume set on the destination system, enter the RALTER command with the additional RACF-protected volumes specified on the ADDVOL operand.

Multiple users per address space

When only one user ID can be associated with an address space, user-related information (ACEE) is anchored from the address space extension block (ASXB). For applications, such as IMS, that permit multiple users per address space, as each user requires an ACEE, and the application can request RACROUTE REQUEST=VERIFY CREATE to return a pointer to the ACEE rather than anchoring it from the ASXB. The application must then keep track of user/ACEE relationships by passing the appropriate ACEE pointer to RACROUTE REQUEST=AUTH and RACROUTE REQUEST=VERIFY (CHANGE or DELETE), thus associating the correct ACEE with the user for whom processing is being done. RACROUTE REQUEST=VERIFY allows the invoker to specify a subpool from which the ACEEs and related storage can be obtained.

The program control option that provides protection for individual load modules does not provide protection for multiple users in an address space. If one user in an address space causes a program to be loaded, another user in the same address space can also execute the program.

Restarting jobs

When a job automatically restarts and returns to a previous checkpoint, RACF repeats user verification and access authorization checking. If the job changed the password on the JOB statement, RACF uses the new password for user verification. But if the PASSWORD command or another job changes the password in the meantime, and JES user identification propagation is not used, RACF detects a password that is not valid and fails the job.

When you resubmit a job for a deferred restart, you should specify your current password on the JOB statement.

An additional consideration exists for tape volumes. If a user is restarting a job with a *deferred* step restart that has the PROTECT parameter on a DD statement, the job does not fail if the tape volume had been defined to RACF before the restart with the PROTECT=YES parameter. On the other hand, an *automatic* step restart with PROTECT specified is successful only if the step abnormally terminated before the tape data set was opened for output and before the tape volume was defined to RACF.

For either an automatic or deferred restart, RACF checks the current access authority at the time of the restart.

Panel driver interface

The panel driver interface provides a way for an ISPF or CLIST programmer to write an application program to invoke the RACF panels without exiting from the current application or facility. After RACF processing is complete, RACF automatically returns to the application from which it was invoked.

ISMF (a part of SMS) uses the panel driver interface for the PROTECT line operator it provides for use on its data set lists under ISPF. If you use the PROTECT line operator you must ensure that the RACF panels, skeletons, and message libraries are allocated whenever ISMF is being used.

For a detailed explanation of the panel driver interface, see *z/OS Security Server RACF Macros and Interfaces*.

REXX RACVAR function

The REXX RACVAR function is a RACF service for REXX execs. It provides information about the running user.

The REXX RACVAR function has four arguments. They are:
- USERID—the user ID that is in the ACEE
- GROUPID—the group name that is in the ACEE
- SECLABEL—the security label that is in the ACEE
- ACEESTAT—the status of the ACEE

Installing the REXX RACVAR function

To execute the REXX RACVAR function, your REXX parameter module must contain an entry for RACF's IRREFPCK directory package, which supports the RACVAR function. For information on REXX parameter modules and how to update and integrate them, see the sections on programming services, function packages, and function directories in *z/OS TSO/E REXX Reference*.

To install the REXX RACVAR function, follow these steps:

Note: Module names, label names, and characteristics specified here are samples derived from current packages and are subject to change. For up-to-date accuracy and completeness, refer to *z/OS TSO/E REXX Reference*.

1. You need to update one or more of the following REXX parameter modules:
   - IRXPARMS (for MVS)
   - IRXTSPRM (for TSO/E)
   - IRXISTRM (for ISPF)

   Locate these modules. You should use the ones that are already installed in your system, but sample copies are available in the SYS1.SAMPLIB library. Their member names are:
   - IRREXX1 for IRXPARMS
   - IRREXX2 for IRXTSPRM
   - IRREXX3 for IRXISTRM
2. In the parameter module, the section for the function package entries is under the header PACKTB_HEADER, which is followed by headers for each level (system, local, and user). The system level is a good place to install RACVAR, although you can also install it in the local or user level.

   The header for the system level is labelled PACKTB_SYSTEM_FIRST. There are two fields under the header for number of entries. For the system level, these fields are PACKTB_SYSTEM_TOTAL and PACKTB_SYSTEM_USED. Increment the count in each field by one. For example if the field is `DC F'2'`, change it to `DC F'3'`.
3. Locate the set of entries pointed to by the header. For the system level, the entries are pointed to by PACKTB_SYSTEM_FIRST. Add another entry here for IRREFPCK. An entry is an 8-byte character constant. Its format should follow the pattern of other entries in the parameter module.
4. Assemble and link-edit the parameter module and place it in SYS1.LINKLIB or any other load module accessible by your system.

Using the REXX RACVAR function

For information on using the REXX RACVAR function, see *z/OS Security Server RACF Macros and Interfaces*.

Chapter 5. RACF remote sharing facility (RRSF)

Overview of the RACF remote sharing facility (RRSF) . . . 120
Understanding the RRSF concepts . . . 120
RRSF nodes and the RRSF network . . . 120
The RRSFDATA class . . . 120
User ID associations . . . 120
Overview of the RRSF function . . . 121
Command direction . . . 121
Password synchronization . . . 122
Automatic direction . . . 123
The RRSF network . . . 128
RRSF nodes . . . 128
Local and remote RRSF nodes . . . 129
Single-system nodes and multisystem nodes . . . 129
Local and remote modes . . . 131
Connections between nodes . . . 132
Operative connections . . . 132
Dormant connections . . . 132
Connection states . . . 132
Workspace data sets . . . 134
Naming conventions for the workspace data sets . . . 134
Defining the workspace data sets . . . 135
Maintaining the workspace data sets . . . 136
Deleting the workspace data sets . . . 137
How a directed command travels through the network . . . 137
Order considerations for directed commands and application updates . . . 140
Defining an RRSF environment . . . 141
Preparing to configure an RRSF network . . . 142
System prerequisites . . . 143
RACF template version considerations . . . 143
RACF dynamic parse version considerations . . . 143
SETROPTS options considerations . . . 144
Class descriptor table considerations . . . 144
Ensuring the security of RACF database information in a network . . . 145
Synchronizing database profiles . . . 145
VTAM and APPC/MVS considerations for an RRSF network . . . 145
Installation exit considerations . . . 149
Considerations for installation-provided code . . . 151
RACF subsystem address space considerations . . . 151
Selecting the main system for a multisystem node . . . 152
Configuring an RRSF network . . . 152
The SET command . . . 152
Listing the attributes of the local node . . . 154
Tracing APPC and IMAGE events . . . 154
Activating and deactivating RRSF functions . . . 154
Specifying a parameter library member to process . . . 155
Specifying a JES node to return output to . . . 155
The TARGET command . . . 155
Defining RRSF nodes to RACF . . . 156
Listing the attributes of target nodes . . . 161
Controlling outgoing requests from the local node . . . 164
Controlling incoming requests from remote nodes . . . 165
Purging a workspace data set . . . 165
Deleting a node . . . 165

© Copyright IBM Corp. 1994, 2004

Reconfiguring a multisystem node 166
The RACF parameter library 168
Customizing and establishing security for RRSF 174
Customizing a remote sharing environment 174
Establishing security for your remote sharing environment 176
Examples of defining a remote sharing environment 176
Configuring nodes in local mode 176
Configuring a two-node network 178
Configuring a multisystem node 179
Configuring two multisystem nodes 181
Monitoring your remote sharing environment 183

This chapter describes aspects of the RACF remote sharing facility (RRSF) that system programmers should be aware of.

Overview of the RACF remote sharing facility (RRSF)

The RACF remote sharing facility (RRSF) exploits the networking services provided by APPC/MVS to extend RACF functionality beyond the single host and shared DASD environments to a network of RRSF nodes capable of communicating with each other. Using the function provided by RRSF, it is possible to administer RACF databases distributed throughout an enterprise from any location in the enterprise.

Understanding the RRSF concepts

Several concepts are central to understanding the RACF remote sharing facility.

RRSF nodes and the RRSF network

The RACF remote sharing facility is built on the concept of a network of RRSF nodes. An *RRSF node* is an MVS system image, or several MVS system images sharing a RACF database, that has been defined as an RRSF node to RACF. Before you can use the functions provided by RRSF, you must configure your MVS system images into a network of RRSF nodes. For detailed information see “The RRSF network” on page 128 and “Configuring an RRSF network” on page 152.

The RRSFDATA class

Profiles in the RRSFDATA class determine which remote sharing functions are available on a node, and which users have access to them. The RRSFDATA class must be active in order for the functions it protects to be available. See “Customizing and establishing security for RRSF” on page 174 for more information on the RRSFDATA class.

User ID associations

Some RRSF functions require a previously established user ID association. A *user ID association* is an association between two user IDs, on the same or different RRSF nodes, that is defined to RACF using the RACLINK command. Typically user ID associations are established between user IDs used by the same person.

There are two types of user ID association: peer and managed. A *peer association* allows either of the associated user IDs to direct commands to the other (see “Command direction” on page 121) and allows the associated user IDs to synchronize their passwords (see “Password synchronization” on page 122). 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 managing user ID can direct commands to the managed ID, but the managed ID cannot direct commands to the managing ID. The user IDs in a managed association cannot synchronize their passwords.

Profiles in the RRSFDATA class control whether user ID associations can be defined, to which nodes they can be defined, and which users can define them. See “Customizing and establishing security for RRSF” on page 174 for more information.

For more information on user ID associations, see *z/OS Security Server RACF General User's Guide* and *z/OS Security Server RACF Security Administrator's Guide*. For information on the RACLINK command, see *z/OS Security Server RACF Command Language Reference*.

Overview of the RRSF function

This section provides an overview of the functions that RRSF provides. For a more detailed description of these functions, see *z/OS Security Server RACF Security Administrator's Guide*.

Command direction

A user logged on to one user ID can issue a RACF command and *direct* that command to run under the authority of the same or another user ID on the same or another RRSF node. This function is referred to as *command direction*. The command runs asynchronously in the RACF subsystem address space, and the output (up to 4096 lines) is returned to the issuing user's RRSFLIST data set. (For more information on the RRSFLIST data set, see *z/OS Security Server RACF General User's Guide* and *z/OS Security Server RACF Security Administrator's Guide*.) Note that when a RACF command runs in the subsystem address space, the accounting information for the command points to the subsystem address space, not the user's address space.

Most RACF commands can be directed. To direct a command, use the AT keyword on the command to specify the RRSF node and user ID the command is to be directed to. (For more information on which commands can be directed, and how to use the AT keyword, see *z/OS Security Server RACF Command Language Reference*.)

Before a user can direct a command to run under another user ID, a user ID association must be established between the user's ID and the other user ID. If a peer association has been established, either user ID can direct commands to the other. If a managed association has been established, the managing user ID can direct commands to the managed user ID, but the managed ID cannot direct commands to the managing ID.

Example: Dan is a user who has two user IDs: user ID DAN on RRSF node MVS01 and user ID DANT on RRSF node MVS02. Either Dan (if he has authorization) or the security administrator can use the RACLINK command to set up a user ID association between DAN on MVS01 and DANT on MVS02. Then, if DAN is logged on to MVS01 and needs to issue a RACF command from his user ID DANT on MVS02, he can do this without logging off from MVS01 by issuing the RACF command on MVS01 and including the AT(MVS02.DANT) keyword to direct the command. RACF sends the command to MVS02 where it runs in the RACF subsystem address space under the authority of the user ID DANT. When the command completes, RACF sends the output back to MVS01 and places it in user ID DAN's RRSFLIST data set.

At times a user might find it useful to direct a command to the user ID and RRSF node he or she is logged on to. No user ID association is required to do this. For example, if Howard is logged on to user ID HOWARD on RRSF node MVS06, he can direct a command to that user ID by specifying the AT(MVS06.HOWARD) or AT(.HOWARD) keyword. The advantages of doing this are:

- The command runs asynchronously in the RACF subsystem address space, instead of synchronously in the user's address space.
- The output is returned in the RRSFLIST data set for user ID HOWARD instead of to the display.

Profiles in the RRSFDATA class control to which nodes command direction is allowed, and which users can direct commands. See "Customizing and establishing security for RRSF" on page 174 for an overview of this control and *z/OS Security Server RACF Security Administrator's Guide* for a detailed discussion.

See "How a directed command travels through the network" on page 137 for details on how RACF processes directed commands and transmits them through the network.

Password synchronization

If password synchronization is enabled between two user IDs, when the password is changed for one of the user IDs, RACF automatically changes the password for the other. The password must be changed in one of the following ways in order for the change to be synchronized:

- By logon processing
- By the PASSWORD command
- By the ALTUSER command
- By an application that uses ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE and supplies the new password in clear text form
- By an application that uses ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE to change both the password and the PASSDATE field or uses the PASSDATA field name to change both fields. This request updates both the password and the passdate information.
- By an application that uses ICHEINTY or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE to change the PASSDATE field or uses the PASSDATA field name to change both fields. This request updates both the password and the passdate information.

Note: If an application uses ICHEINTY, RACROUTE REQUEST=EXTRACT, or RACXTRT to change only the PASSDATE field, the password does not change.

Password changes made in the following ways are not synchronized:

- By the ADDUSER command
- By an application that encrypts the new password before specifying it on an ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE because encryption methods can differ on different systems

Use the PWSYNC and NOPWSYNC keywords on the RACF SET command to activate and deactivate password synchronization. Use the OUTPUT and NOTIFY subkeywords of SET PWSYNC to specify which users will be notified of results and receive output from password synchronization requests. For more information, see "The SET command" on page 152.

Password synchronization is enabled between two user IDs by creating a peer user ID association between the two IDs that specifies password synchronization.

Profiles in the RRSFDATA class control who can define user ID associations with password synchronization enabled, whether password synchronization occurs on an RRSF node, and for which users. See “Customizing and establishing security for RRSF” on page 174 for more information.

Automatic direction

Automatic direction allows you to have RACF automatically direct updates made to the RACF database on an RRSF node to one or more other RRSF nodes. If profiles on two or more RRSF nodes are already synchronized, you can use automatic direction to have RACF automatically keep the profiles synchronized. RACF provides the following types of automatic direction:

- Automatic command direction, discussed in “Automatic command direction”
- Automatic password direction, discussed in “Automatic password direction” on page 124
- Automatic direction of application updates, discussed in “Automatic direction of application updates” on page 125

Profiles in the RRSFDATA class control which types of automatic direction are active. The RACF SET command activates and deactivates automatic direction. For more information on controlling automatic direction, see “Customizing and establishing security for RRSF” on page 174.

Automatic command direction: Automatic command direction extends the function of command direction to automatically direct commands issued on an RRSF node to one or more other RRSF nodes. If profiles on two or more RRSF nodes are already synchronized, you can use automatic command direction to have RACF automatically keep the profiles synchronized with respect to RACF commands. That is, if a user issues a RACF command on an RRSF node, either as a TSO/E command or as an operator command, after the command successfully runs on that node (completes with a return code less than or equal to 4), RACF automatically directs that command to the other nodes.

Commands that are issued on a node, or that are directed to a node using the AT keyword, can be automatically directed from that node. Commands that are directed to a node using the ONLYAT keyword, or that are automatically directed to a node, are not automatically directed from that node.

Unlike command direction, automatic command direction does not require user ID associations. Instead, automatic command direction assumes that if the same user ID exists on two different nodes, those user IDs belong to the same person. When a command is directed automatically, it runs under the authority of the user ID that issued it, but on another node, it runs with the authority defined for that user ID on the other node. For example, if an RRSF network was set up to automatically direct all commands from node MVSX to MVSY, and user ID BOBH issued an ADDUSER command on node MVSX, RACF would run the command on node MVSX under the authority defined for BOBH on MVSX. If it completed successfully RACF would then automatically direct the command to run on node MVSY, under the authority defined for BOBH on MVSY.

Unlike command direction, automatic command direction does not require that the command issuer specify the AT keyword, or take any other explicit action to make the automatic command direction occur. In fact, depending on how you choose to set up automatic command direction, it can be transparent to the command issuer. In the preceding example, BOBH might not even realize that the command he issued on MVSX also ran on MVSY.

When commands are automatically directed, they always run on the local node before being directed. In contrast, when commands are directed using the AT or ONLYAT keywords, they do not run on the local node unless they are explicitly directed to run on the local node by the AT or ONLYAT keyword.

Profiles in the RRSFDATA class control which commands are automatically directed, and to which nodes. See “Customizing and establishing security for RRSF” on page 174 for more information. You can choose to have commands automatically directed based on the command name, the profile class to which the command is related (USER, GROUP, DATASET, or any general resource class) or who issued the command.

After you define your RRSFDATA profiles to set up automatic command direction and activate the RRSFDATA class, use the AUTODIRECT and NOAUTODIRECT keywords on the RACF SET command to activate and deactivate automatic command direction. Use the OUTPUT and NOTIFY keywords on the SET command to specify which users will be notified of results and receive output from automatically directed commands. See “The SET command” on page 152 for more information.

RACF provides the ONLYAT keyword for situations where automatic command direction is active and, due to a failure, an update is made on one node but not on another, causing an inconsistency between the RACF databases. A user with SPECIAL authority can issue a command to correct the inconsistency, specifying the ONLYAT keyword to cause the command to be run only on the node specified. Automatic command direction does *not* occur for a command that specifies the ONLYAT keyword.

Automatic password direction: Automatic password direction extends the function of password synchronization to automatically synchronize passwords without requiring user ID associations. An installation using automatic direction to synchronize user profiles can optionally request activation of automatic password direction. The password must be changed in one of the following ways in order for automatic password direction to be in effect:

- By logon processing
- By an application that uses ICHEINTY or RACROUTE REQUEST=VERIFY or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE and supplies the new password in clear text form
- By an application that uses ICHEINTY or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE to change the PASSDATE field or uses the PASSDATA field name to change both fields. This request updates both the password and the passdate information.

Password changes made in the following ways are not eligible for automatic password direction:

- By the ADDUSER command
- By the ALTUSER command
- By the PASSWORD command
- By an application that encrypts the new password before specifying it on an ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE because encryption methods can differ on different systems

Note: If an application uses ICHEINTY, RACROUTE REQUEST=EXTRACT, or RACXTRT to change only the PASSDATE field, the password does not change.

Like automatic command direction, automatic password direction assumes that if the same user ID exists on two different nodes, those user IDs belong to the same person. For example, if automatic password direction is active between nodes ATLANTA and RALEIGH for all users, and user ID MARYM exists on nodes ATLANTA and RALEIGH, if Mary changes her password for user ID MARYM on node ATLANTA, after RACF changes the password for MARYM on node ATLANTA it makes the same change to MARYM on node RALEIGH. Mary does not have to create a user ID association between her ATLANTA and RALEIGH user IDs.

Use the AUTOPWD and NOAUTOPWD keywords on the RACF SET command to activate and deactivate automatic password direction. Use the OUTPUT and NOTIFY subkeywords of SET AUTOPWD to specify which users will be notified of results and receive output from automatic password direction. For more information, see "The SET command" on page 152.

Profiles in the RRSFDATA class control for which users password changes are automatically directed, and to which nodes. See "Customizing and establishing security for RRSF" on page 174 for more information.

Automatic direction of application updates: When automatic direction of application updates is active, RACF automatically directs application updates made by the following macros to selected remote nodes:

- RACROUTE REQUEST=DEFINE or RACDEF
- RACROUTE REQUEST=EXTRACT,TYPE=REPLACE or RACXTRT specifying TYPE=REPLACE
- ICHEINTY ADD, ALTER, DELETE, DELETEA, and RENAME requests

If profiles on two or more RRSF nodes are already synchronized, you can use automatic direction of application updates to keep the profiles synchronized with respect to application updates.

Note: Automatic direction of application updates does not propagate password related updates on the same request used to change a password. For example, when you use a RACROUTE REQUEST=EXTRACT or ICHEINTY request to change PASSWORD, PASSDATE, and INSTDATA information or if you update PASSWORD, password history, and INSTDATA information, automatic direction of application updates propagates only the INSTDATA changes. Automatic password direction propagates the other changes in conjunction with the password synchronization request. When you request an update that does not include a password change, automatic direction of application updates propagates all fields except PASSDATE.

RACF directs an application update only after the update has successfully completed on the node on which the application is executing.

Not all RACROUTE REQUEST=DEFINE and RACDEF requests update the RACF database, and RACF does not automatically direct requests that do not update the database. RACROUTE REQUEST=DEFINE and RACDEF are not automatically directed if:

- ENVIR=VERIFY is specified
- RACFIND=NO is specified and DSTYPE=T is not specified

ICHEINTY Macros Issued By Other Macros: RACROUTE REQUEST=VERIFY and RACINIT update the RACF database by issuing ICHEINTY macros. When automatic direction of application updates is active, RACF automatically directs the following ICHEINTY requests made by RACROUTE REQUEST=VERIFY and RACINIT:

- The ICHEINTY request that sets the revoke flag in the user profile when a user is being revoked due to inactivity or password attempts that are not valid
- The ICHEINTY request that increments the revoke count when a user enters a password that is not valid
- The ICHEINTY request that resets the revoke count to 0 when a user enters a valid password, if the revoke count for the user was nonzero before the update was made

Automatic direction of the ICHEINTY request that RACROUTE REQUEST=VERIFY issues to change the password in the user's profile is controlled by automatic password direction, and not by automatic direction of application updates. Automatic direction of the ICHEINTY request that sets the revoke count to zero is controlled by automatic direction of application updates. Note that if the same ICHEINTY both changes the user's password and sets the revoke count to zero, RACF generates two RRSF requests, one from automatic password direction for the password change, and one from automatic direction of application updates for the setting of the revoke count.

When a RACROUTE REQUEST=DEFINE, a RACDEF, a RACROUTE REQUEST=EXTRACT, or a RACXTRT issues an ICHEINTY macro, RACF does not direct the ICHEINTY request separately.

Interaction with automatic command direction: Automatic command direction determines whether a RACF command is directed. If a command issues a RACROUTE or ICHEINTY macro, the macro is not directed by automatic direction of application updates.

Activation and deactivation: Use the AUTOAPPL and NOAUTOAPPL keywords on the RACF SET command to activate and deactivate automatic direction of application updates. Use the OUTPUT and NOTIFY subkeywords of SET AUTOAPPL to specify which users will be notified of results and receive output from automatically directed application updates. For more information, see "The SET command" on page 152.

Controlling Which Updates Are Directed: Profiles in the RRSFDATA class control which application updates are automatically directed to which nodes. See "Customizing and establishing security for RRSF" on page 174 for more information.

Considerations for the DATASET class: There are special considerations for profiles in the DATASET class. The creation, renaming, and deletion of discrete profiles in the DATASET class is closely tied to the data set itself. For example:

- If a discrete data set profile is created, the RACF indicator bit for the data set must be turned on in the VTOC or catalog entry in order for RACF to find the profile during authority checking.
- If a discrete data set profile is renamed and the data set itself is not renamed, the discrete profile is no longer used to protect the data set.
- If a discrete data set profile is deleted and the data set itself is not deleted, the data set might then be protected by an existing generic profile.

Because RACROUTE REQUEST=DEFINE and RACDEF manipulate only the RACF profiles, and not the data sets or their RACF indicator bits, you should not

have changes made by these macros directed automatically if your installation uses any discrete data set profiles. The exception is when two systems share DASD, but not the RACF database, resulting in two copies of the profile for a DASD data set. This situation might occur during migration to the Parallel Sysplex. For example, if you have 4 systems sharing DASD and the RACF database, you might want to begin your migration to the Parallel Sysplex by putting just two systems in the Parallel Sysplex. The other two systems cannot continue to share the RACF database with the two in the Parallel Sysplex, so you might have two copies of the RACF database, and keep them synchronized using RRSF. In this case, if you create a data set on a system in the Parallel Sysplex, you want the profile to be created on both copies of the RACF database, to allow you to access the data set from the systems that are not yet in the Parallel Sysplex.

There are similar considerations for data sets that reside on tape. In addition, if the TAPEVOL class is active and TVTOCs are used, a RACROUTE REQUEST=DEFINE for the DATASET class or a RACDEF with DSTYPE=T for the DATASET class results in updates to both the DATASET and TAPEVOL classes. Even if RACFIND=NO is specified, which prevents an update to the DATASET class profile, the TAPEVOL profile is updated. If you have set up your RRSFDATA profiles to exclude the TAPEVOL class from automatic direction of commands and automatic direction of application updates, you might also want to exclude tape data sets from automatic direction of application updates.

Exit considerations: A RACROUTE REQUEST=DEFINE request can pass information to the RACROUTE REQUEST=DEFINE installation exits via the INSTLN and ACCLVL keywords. The RACROUTE REQUEST=DEFINE installation exits, ICHRDX01 and ICHRDX02, must inform RACF whether or not to automatically direct this parameter information, and how much of it to automatically direct. For more information, see “RACROUTE REQUEST=DEFINE exits” on page 283.

For information on when application update requests issued from RACF exits are and are not propagated by automatic direction, see “Installation exit considerations” on page 149. For information on when application update requests issued from SAF exits are and are not propagated by automatic direction, see *z/OS Security Server RACROUTE Macro Reference*.

Releases that do not support automatic direction of application updates: Automatic direction of application updates is not supported on releases of OS/390 Security Server earlier than Release 3 or on RACF Versions 1 and 2. If you configure automatic direction of application updates to a system that does not support it, RACF issues an error message for the first application update it attempts to direct to that system, and discards subsequent updates without additional messages.

IBM products that can be affected: Table 4 lists some IBM products that use RACROUTE or ICHEINTY to update the RACF database. Note that ADSP and PROTECT=YES (JCL) can cause database updates. You can use automatic direction of application updates to synchronize changes these products make to RACF databases.

Table 4. Some IBM products that use RACROUTE or ICHEINTY to update the RACF database

Product	Class and Type of Profile
Application System	Creation and deletion of profiles in the RACF class used by AS, usually DASNAMES.

Table 4. Some IBM products that use RACROUTE or ICHEINTY to update the RACF database (continued)

Product	Class and Type of Profile
DFSMSdfp™	Creation and deletion of discrete profiles in class DATASET when ADSP or PROTECT=YES is specified. Update of discrete profiles in class TAPEVOL with TVTOC, creation of discrete automatic TAPEVOL profiles with TVTOC.
DFSMShsm™	Update of discrete profiles in class TAPEVOL for tape volumes managed by DFSMShsm.
DFSMSrmm™	Creation and deletion of discrete profiles in class TAPEVOL for tape volumes managed by DFSMSrmm.
NetView Access Services	Creation and deletion of profiles in class NVASAPDT.
RACF	Maintenance of the revoke count and revoke indicator fields in the USER profile. Registration and deregistration of digital certificates in class DIGTCERT using RACDCERT or initACEE.

The RRSF network

An RRSF network is a collection of RRSF nodes in a network using APPC/MVS as the network transport mechanism. Figure 10 illustrates an RRSF network.

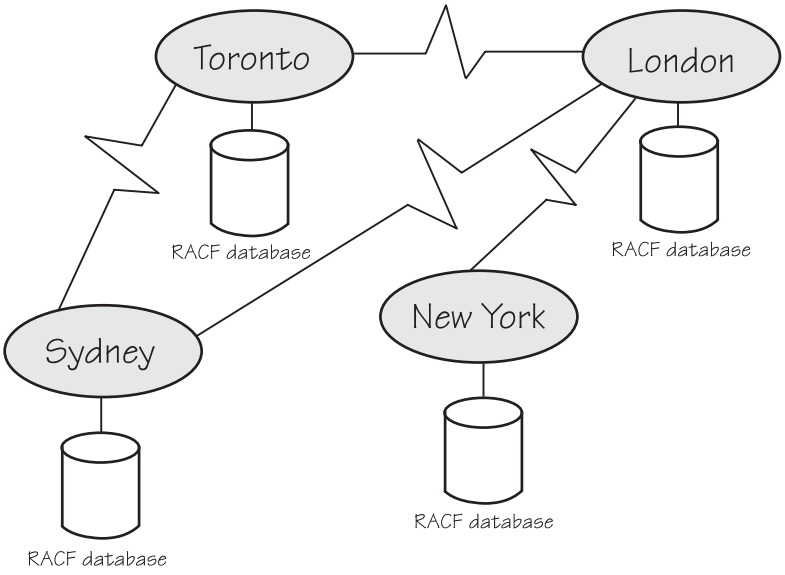


Figure 10. An RRSF network

RRSF nodes

An RRSF node is an MVS system image, or a group of MVS system images sharing a RACF database, that has been defined as an RRSF node to RACF by a TARGET command. See “Defining RRSF nodes to RACF” on page 156 for details on defining RRSF nodes with the TARGET command. An MVS system image must meet the following requirements to be defined as an RRSF node:

- MVS/ESA 4.3 or later, z/OS, or OS/390 is installed.

- RACF 2.2 or the RACF component of the z/OS or OS/390 SecureWay Security Server is installed.
- The RACF subsystem address space is active.

In order to direct commands or application updates from one MVS system image to another, or synchronize passwords between two MVS system images, both of the system images must first be defined to RACF as RRSF nodes that can communicate with each other.

Local and remote RRSF nodes

The terms local and remote can be useful when discussing RRSF nodes. The *local* node is the node whose viewpoint you are speaking from. Its *remote* nodes are the other nodes in the network with which it communicates. For example, in the network shown in Figure 10 on page 128, from the Toronto node's point of view, the Toronto node is the local node and the Sydney and London nodes are remote nodes. The Toronto node cannot communicate with the New York node. From the London node's point of view, the London node is the local node and the Toronto, Sydney, and New York nodes are remote nodes. From the Sydney node's point of view, the Sydney node is the local node and the Toronto and London nodes are remote nodes. The Sydney node cannot communicate with the New York node. From the New York node's point of view, the New York node is the local node and the London node is a remote node. The New York node cannot communicate with the Sydney and Toronto nodes.

Single-system nodes and multisystem nodes

An RRSF node can be either a single-system node or a multisystem node. A *single-system RRSF node* consists of one and only one MVS system image. A *multisystem RRSF node* consists of multiple MVS system images that share a RACF database.

For a multisystem RRSF node, you designate one of the MVS system images to be the *main system*. The main system receives most of the RRSF communications sent to the node. The other systems in the node are known as *nonmain systems*. Figure 11 on page 130 shows an RRSF network containing a single-system node and a multisystem node.

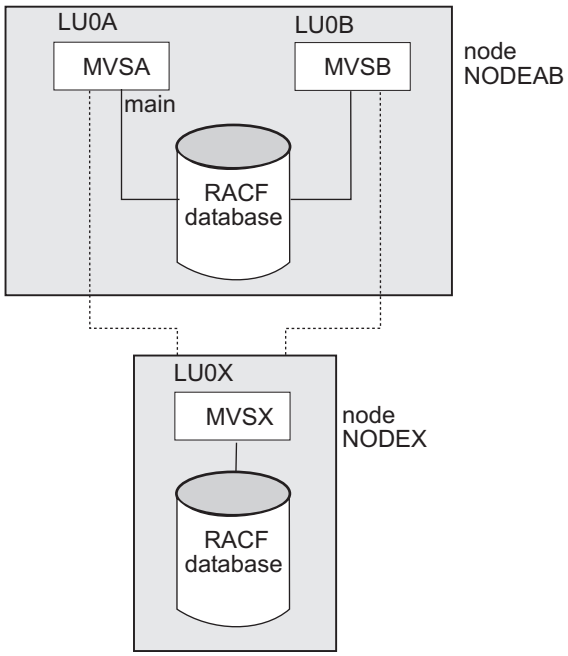


Figure 11. An RRSF network containing a single-system node and a multisystem node. NODEAB is a multisystem node. NODEX is a single system node.

Main systems in a multisystem RRSF node can send RRSF requests to main systems on remote multisystem RRSF nodes, and to single-system RRSF nodes. In addition, when main systems receive requests from remote systems (main or nonmain), they send output and notifications back to the system that originated the request.

Nonmain systems in a multisystem RRSF node can send RRSF requests to main systems on remote multisystem RRSF nodes, and to single-system RRSF nodes. They cannot send RRSF requests to other remote nonmain systems, or to other local systems (nonmain or main).

Most RRSF communications sent to the multisystem RRSF node are received by the main system, including:

- All commands directed to the multisystem node
- All RACLINK requests sent to the multisystem node
- All password changes sent to the multisystem node
- All output and notifications from automatically directed commands and application updates

The following types of RRSF communications can be received by any system in a multisystem node:

- Output and notifications from commands that were directed via the AT or ONLYAT keywords. These are returned to the system on which the directed command was issued.
- Notifications from RACLINK commands. These are returned to the system on which the RACLINK command was issued.
- Output from password changes when automatic password direction is used. These are returned to the system on which the password was changed.

When the local node is a multisystem node, a system in the node is referred to as either the local system or a local peer system. The *local system* is the system in the

local multisystem RRSF node whose SYSNAME (defined by the TARGET command) matches the current CVTSNAME. A *local peer system* is a system in the local multisystem RRSF node whose SYSNAME (defined by the TARGET command) does not match the current CVTSNAME. All of the systems in a local multisystem RRSF node are referred to as *member systems* of the node.

Figure 12 shows an RRSF network containing two multisystem nodes, NODEXY and NODEAB, and one single-system RRSF node, NODEC. Multisystem node NODEXY contains two systems, MVSX and MVSY. Multisystem node NODEAB contains two systems, MVSA and MVSB.

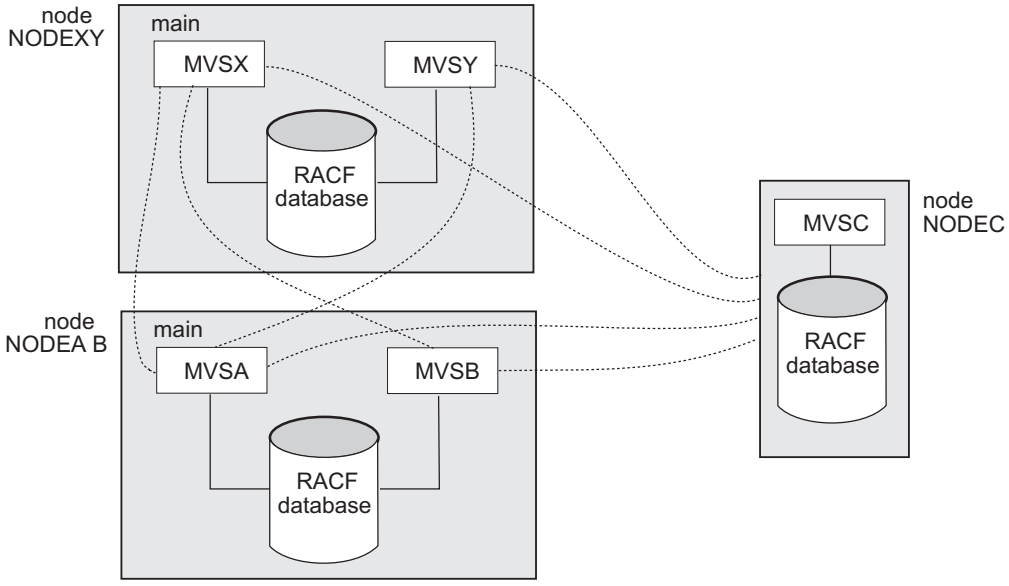


Figure 12. An RRSF network with two multisystem nodes and one single-system node

On NODEAB, from the perspective of system MVSB:
- NODEAB is the *local RRSF node*.
- MVSA and MVSB are the *member systems* of node NODEAB.
- MVSB is the *local system*.
- MVSA is a *local peer system*.
- MVSA is the *local main system*.
- NODEXY and NODEC are *remote RRSF nodes*.
- MVSX is a *remote main system*.
- MVSY is a *remote nonmain system*.

Local and remote modes

An RRSF node can operate in either local mode or remote mode.

When an RRSF node operates in *local mode*, it is not configured to communicate with other RRSF nodes. A node operating in local mode provides limited remote sharing functions:

- Users with multiple user IDs on the node can synchronize passwords between those user IDs.
- Users with multiple user IDs on the node can direct commands to run under the other user IDs.
- Users can direct commands to run in the RACF subsystem on the local node.

When an RRSF node operates in *remote mode*, it is configured to communicate with other RRSF nodes. A node operating in remote mode provides the full power of the RACF remote sharing facility to perform RACF functions across a network.

Connections between nodes

Two RRSF nodes are said to be *logically connected* when they are configured to communicate via APPC/MVS, their RACF subsystem address spaces are active, and they have been defined to RACF as RRSF nodes that can communicate with each other.

At a high level, there are two types of connections between nodes: operative and dormant. At a lower level, the connection between two nodes can be in any one of a number of states, discussed in "Connection states."

Operative connections

When a node's connection to a remote node is fully *operative* (in the operative active state, see Table 5 on page 133), outgoing requests from the node are sent immediately to the remote node. A connection goes through several operative states, described in Table 5 on page 133, in the process of becoming operative active, and in these states RACF saves outgoing requests from the node in a workspace data set until the connection becomes operative active.

Use the TARGET command to:

- Define a connection to another node
- Request that a connection be made operative

See "Controlling outgoing requests from the local node" on page 164 for more information on how to request that a connection be made operative.

Dormant connections

When a node's connection to a remote node is *dormant*, RACF saves outgoing requests to the remote node in a workspace data set until the connection becomes operative. See "Workspace data sets" on page 134 for information on workspace data sets.

Use the TARGET command to:

- Define a connection to another node
- Request that a connection be made dormant

If the connection is operative when you issue a TARGET command to make it dormant, the other node detects that the connection has been made dormant. The other node then saves all further requests for the node that issued the TARGET in a workspace data set.

There might be times when you need to make a connection dormant in order to perform a function. For example, to delete the connection with a node, you must first make the connection with that node dormant.

See "Controlling outgoing requests from the local node" on page 164 for more information on how to define a connection as dormant.

Connection states

While at a high level there are two types of connections between nodes, operative and dormant, at a lower level the connection between two nodes can be in any one of a number of states. Table 5 on page 133 shows the states that can exist for a connection between two nodes. You might see references to these states in error

messages. These states also appear in the output from a TARGET LIST command, showing the status of the connections from the perspective of the local node. See “Listing the attributes of target nodes” on page 161 for more information on TARGET LIST. For further details on connection states and transitions between them, refer to *z/OS Security Server RACF Diagnosis Guide*.

Table 5. Connection States between Nodes

Name	Abbreviation	Description
operative pending connection	O-P-C	The local node has requested that the connection be activated and is attempting to activate the conversation. The local node has not yet received a confirmation that the remote node will accept the connection.
operative pending verification	O-P-V	The local node’s request for a conversation has been accepted. The two nodes are communicating and evaluating information they have exchanged to determine if they are compatible. If the two nodes are not compatible, both nodes will remain in the operative pending verification state.
operative active	O-A	The connection between two nodes is active. The two nodes have verified that they can communicate with each other and that they are compatible with each other.
operative in error	O-E	The local system has lost the connection with the remote node. At one time a connection had been successfully established.
dormant by local request	D-L	The local node’s connection with a remote node has been made dormant by an operator issuing a TARGET DORMANT command.
dormant by remote request	D-R	The local node has detected that the connection to the remote node has been made dormant by the remote node or the connection between the local and remote nodes has not been defined on the remote node.
dormant by mutual request	D-B	The local and remote nodes have both requested the connection be dormant by an operator issuing a TARGET DORMANT command on each system.
dormant in error	D-E	The local node is dormant and a failure is experienced while saving RRSF requests for later processing.
defined	DEF	TARGET information has been defined, but no conversation occurs. This state occurs: • Between member systems of a multisystem node. Systems in a multisystem node do not communicate with each other. • Between a local nonmain system and a nonmain system on a remote multisystem node. Nonmain systems of multisystem nodes can communicate with single-system nodes and with the main systems of multisystem nodes, but they do not communicate with nonmain systems of other multisystem nodes.

Table 5. Connection States between Nodes (continued)

Name	Abbreviation	Description
not defined (initial)	???	No connection has been established to the node due to insufficient configuration information, or because a TARGET OPERATIVE or TARGET DORMANT command has not been issued for the node.

Workspace data sets

Workspace data sets are VSAM data sets that RACF uses to temporarily hold data that RACF is sending from one node to another. RACF deletes data from the workspace data sets when it receives confirmation that the data has been successfully processed at the receiving node. See "How a directed command travels through the network" on page 137 for details about when RACF saves data to and deletes data from the workspace data sets.

RACF uses two workspace data sets, the INMSG data set and the OUTMSG data set, for the local node and for each of its remote nodes.

- **INMSG**

  The INMSG data set is used to temporarily hold requests that are being sent to the local node from itself or a remote node, such as:
  - Commands directed to the local node
  - Output from RACF commands, application updates, and password changes that were directed to a remote node

- **OUTMSG**

  The OUTMSG data set is used to temporarily hold requests that are being sent to a target node, such as:
  - Commands, application updates, and password changes directed from the local node
  - Output to be returned to another node

To protect RACF data from casual viewing while it is in the workspace data sets, RACF masks the data using the Commercial Data Masking Facility (CDMF) algorithm.

Naming conventions for the workspace data sets

The naming convention for the workspace data sets created on a node as a result of a TARGET LOCAL command is:

prefix.sysname_or_wdsqual.ds_identity

where:

prefix Is a value you specify with the PREFIX keyword on the TARGET command

sysname_or_wdsqual
: Is the system name if a WDSQUAL value is not specified on the TARGET command. The SYSNAME must match the value in the CVTSNAME field for the system it identifies. If the WDSQUAL value is specified on the target command, that value is used instead of the SYSNAME.

ds_identity
: Is either INMSG or OUTMSG

The naming convention for the workspace data sets for remote connections is:

prefix.local_luname.remote_luname_or_wdsqual.ds_identity

where:

prefix Is a value you specify with the PREFIX keyword on the TARGET command

local_luname
Is the LU name of the local node

remote_luname_or_wdsqual
Is the LU name of the remote node if a WDSQUAL value is not specified on the TARGET command. If the LU name for a node is a qualified name in the form *netid.luname*, RACF uses only the second part of the qualified LU name in the names of the workspace data sets. If a WDSQUAL value is specified on the TARGET command, RACF uses that value instead of the LU name of the remote node.

ds_identity
Is either INMSG or OUTMSG

The prefix defined for each member system of a multisystem node must be the same.

Example: In the RRSF network shown in Figure 25 on page 179, the following workspace data sets are created on system MVSA:

```
SYS1.MVSA.INMSG
SYS1.MVSA.OUTMSG
SYS1.LU0A.LU0X.INMSG
SYS1.LU0A.LU0X.OUTMSG
```

The following workspace data sets are created on system MVSB:

```
SYS1.MVSB.INMSG
SYS1.MVSB.OUTMSG
SYS1.LU0B.LU0X.INMSG
SYS1.LU0B.LU0X.OUTMSG
```

The following workspace data sets are created on system MVSX:

```
SYS1.MVSX.INMSG
SYS1.MVSX.OUTMSG
SYS1.LU0X.LU0A.INMSG
SYS1.LU0X.LU0A.OUTMSG
SYS1.LU0X.LU0B.INMSG
SYS1.LU0X.LU0B.OUTMSG
```

Defining the workspace data sets

There are two methods you can use to define the workspace data sets:

- Let RACF allocate the VSAM data sets for you.
- Preallocate the VSAM data sets yourself.

Make sure that the RACF subsystem user ID has the authority to create and access the workspace data sets. The recommended way to do this is to define the subsystem user ID as trusted or privileged. See “Assigning a user ID to the RACF subsystem” on page 74 for more information.

If You choose to preallocate the VSAM data sets: SYS1.SAMPLIB member IRRSRRSF contains a sample member RRSFALOC with sample JCL to define the VSAM workspace data sets. Use the PREFIX keyword on the TARGET command to identify the prefix you use to RACF.

RACF might delete data sets you have preallocated. See “Deleting the workspace data sets” on page 137.

If You choose to let RACF allocate the VSAM data sets: You define the workspace data sets using the TARGET command when you define a target node. The WORKSPACE keyword defines the attributes of the workspace data sets, and the PREFIX keyword defines their prefix. See “Defining RRSF nodes to RACF” on page 156 for more information.

You specify the volume on which the workspace data sets reside on the TARGET command. Select a volume that has sufficient room to allow for the expansion of the VSAM data sets.

RACF uses the information you provide on the WORKSPACE keyword whenever it has to allocate a new workspace data set. If you issue a TARGET command to make a node operative or dormant and the workspace data sets do not exist, RACF allocates new data sets using the workspace information defined at that time. However, if you issue a TARGET command with new values for these keywords, and the workspace data sets already exist, RACF does *not* reallocate the data sets. If you issue a TARGET LIST command, it shows the new values you provided on the TARGET command, which are not the values actually in effect. For example, if you issue a TARGET DELETE command to delete a node, and there are still records in a workspace data set for that node, RACF does not delete the data set. If you later issue a TARGET command to that node to reconnect to it, and the workspace data set still exists and is cataloged, and its name matches the name RACF generates, RACF reuses the existing data set. If you changed the WORKSPACE keywords on the TARGET command you issued to reconnect, those values do not take effect, but they are shown if you do a TARGET LIST for the node. The new values take effect the next time RACF allocates a new workspace data set.

Size guidelines for the workspace data sets: Specify the size of a workspace data set with the FILESIZE keyword on the TARGET command. Some guidelines to follow are:

- If your RACF administrators submit large batch jobs containing hundreds or thousands of commands, you’ll need larger workspace data sets.
- If you expect nodes in your RRSF network to be dormant for long periods of time, you’ll need larger workspace data sets.
- If one of your RRSF nodes takes an unusually long time to IPL, nodes that communicate with it will need larger workspace data sets.
- If you plan to have a node down for a few hours for a hardware upgrade, nodes that communicate with it will need larger workspace data sets.

Maintaining the workspace data sets

It is important to prevent the workspace data sets from filling up. If they do fill up, commands might be rejected and database inconsistencies might occur.

Determining how full the workspace data sets are: You can monitor how full the workspace data sets are for an RRSF node by periodically issuing a TARGET LIST command for the node to see how many records are used and how many extents exist.

Increasing the size of the workspace data sets: For information on how to increase the size of the workspace data sets, see “Recovering When the Workspace Data Sets Fill Up” on page 329.

Deleting the workspace data sets

When the workspace data sets are empty, a TARGET NODE(*nodename*) DELETE command deletes the data sets (in addition to removing the node from your configuration). The data sets are deleted regardless of whether RACF allocated them or you preallocated them yourself.

How a directed command travels through the network

Figure 13 on page 138 illustrates the journey of a directed RACF command through an RRSF network. The network contains two RRSF nodes, NODEA and NODEB. The LU name for NODEA is LUA, and the LU name for NODEB is LUB. User ID PAT on NODEA has a peer user ID association defined with user ID PATL on NODEB. User ID PATL is defined on NODEB and has authorization to delete user DUDLEY. The prefix for the workspace data sets is RRSF.QUEUES for both NODEA and NODEB.

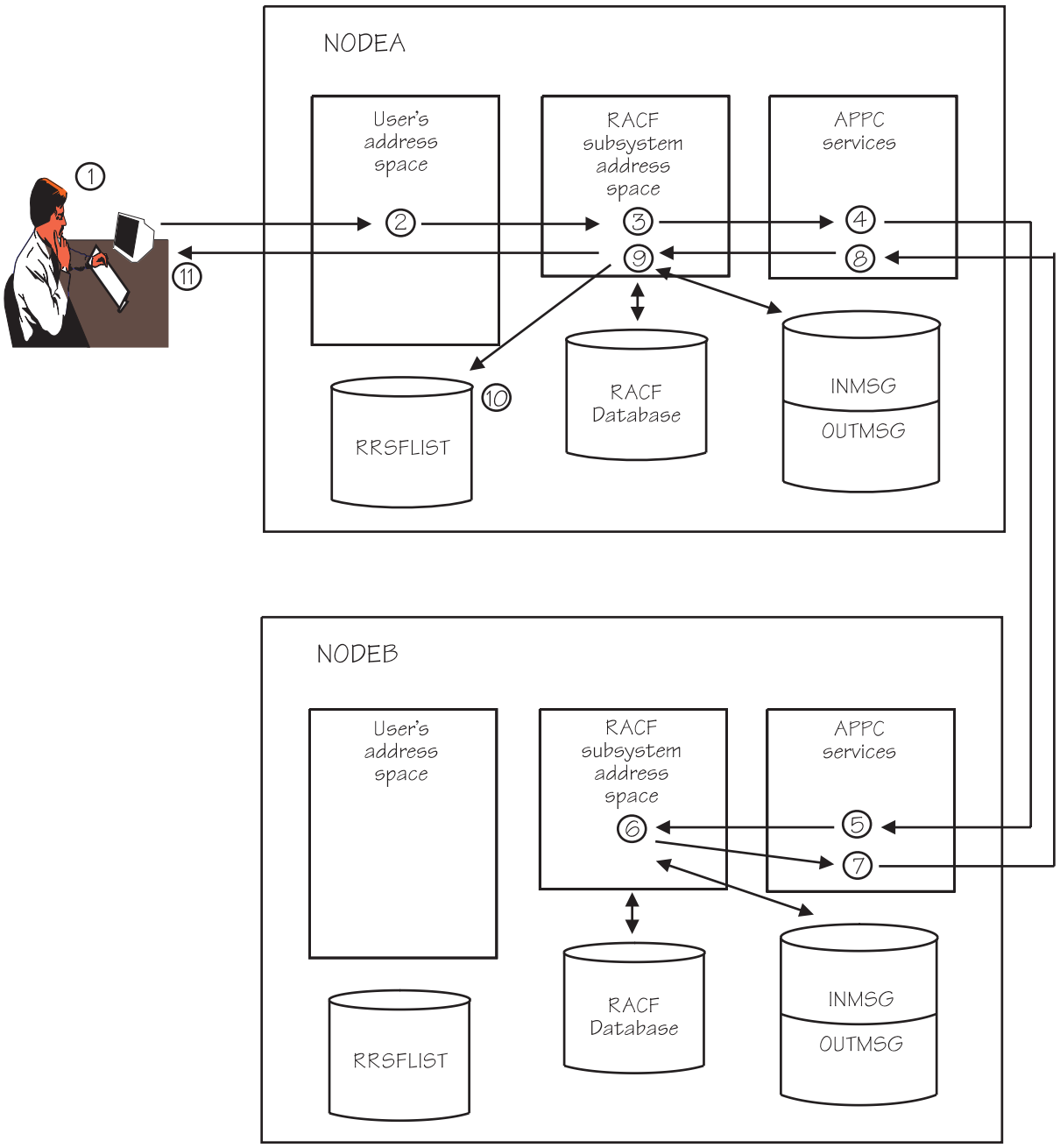


Figure 13. A directed command traveling through an RRSF network

A directed command is processed as follows:

1. User PAT on NODEA issues the command `DELUSER DUDLEY AT(NODEB.PATL)`.
2. RACF on NODEA determines that the command is a directed command, and verifies that:
   a. User ID PAT on NODEA has an association defined with user ID PATL on NODEB.
   b. User ID PAT on NODEA is authorized (via an RRSFDATA profile) to direct commands to NODEB.

   If both conditions are met, RACF sends the command to the RACF subsystem address space.

3. The RACF subsystem address space processes the command as follows:
   a. RACF on NODEA masks the command using the Commercial Data Masking Facility (CDMF) algorithm implemented with a fixed key. The purpose of this masking is to protect against inadvertent casual viewing of the data while it is in the workspace data sets and during transmission. (This protection supplements the protection that classes such as APPCSERV and APPCLU provide to the data during transmission, and that RACF DATASET authorization provides to the data while it is in the workspace data sets.)
   b. RACF on NODEA saves a copy of the masked command in the OUTMSG workspace data set, RRSF.QUEUES.LUA.LUB.OUTMSG.
   c. If the connection between NODEA and NODEB is operative, RACF on NODEA passes the command to APPC/MVS. If the connection is dormant, RACF waits for it to become operative before passing the command to APPC/MVS.
4. APPC/MVS transmits the masked command from NODEA to NODEB.
5. APPC/MVS passes the masked command to RACF on NODEB.
6. RACF processes the command as follows:
   a. RACF on NODEB saves a copy of the masked command in the INMSG workspace data set RRSF.QUEUES.LUB.LUA.INMSG.
   b. RACF on NODEB notifies RACF on NODEA via APPC/MVS that the command has been written to the INMSG workspace data set.
   c. RACF on NODEA deletes the command from the OUTMSG workspace data set, RRSF.QUEUES.LUA.LUB.OUTMSG. Note that the deletion does not occur until after the command is received on NODEB, to prevent loss of the command if transmission errors occur, thus ensuring data integrity.
   d. RACF on NODEB unmasks the masked command.
   e. RACF on NODEB verifies that user ID PATL exists on NODEB and that PATL on NODEB has an association with PAT on NODEA.
   f. RACF on NODEB runs the command in the RACF subsystem address space with the authority of PATL, and deletes user DUDLEY.
   g. RACF on NODEB masks the command output using the CDMF algorithm.
   h. RACF on NODEB saves a copy of the masked output in the OUTMSG workspace data set RRSF.QUEUES.LUB.LUA.OUTMSG.
   i. If the connection between NODEA and NODEB is operative, RACF on NODEB passes the masked output to APPC/MVS. If the connection is dormant, RACF waits for it to become operative before passing the command output to APPC/MVS.
7. APPC/MVS transmits the masked command output from NODEB to NODEA.
8. APPC/MVS passes the masked command output to RACF on NODEA.
9. RACF on NODEA saves a copy of the masked command output in the INMSG workspace data set RRSF.QUEUES.LUA.LUB.INMSG. RACF on NODEA notifies RACF on NODEB via APPC/MVS that the output has been received. RACF on NODEB deletes the command output from the OUTMSG workspace data set RRSF.QUEUES.LUB.LUA.OUTMSG.
10. RACF on NODEA unmasks the command output, stores it in user ID PAT's RRSFLIST data set, and deletes the saved copy in the INMSG workspace data set RRSF.QUEUES.LUA.LUB.INMSG.
11. RACF on NODEA uses SEND to notify user ID PAT that the command has completed.

If user PAT on NODEA had originally issued the command `DELUSER DUDLEY ONLYAT(NODEB.PATL)`, using the ONLYAT keyword instead of the AT keyword, the processing for the command would have been the same, with the following exceptions:

- Step 2b on page 138 would have been replaced by a check that user ID PAT on NODEA had SPECIAL authority.
- Step 6e on page 139 would have additionally checked whether user ID PATL on NODEB had SPECIAL authority.

Order considerations for directed commands and application updates

RACF's APPC/MVS server insures that RRSF requests directed to a remote node are executed by the remote node in the same order that they were issued. However, when multiple users are directing commands to a remote node, commands directed by different users might not be received in the order they were issued, and might not be executed in the order they are received. Similarly, if multiple users are taking actions that cause applications to issue update requests that RACF is directing to a remote node, updates directed for different users might not be received in the order they were issued, and might not be executed in the order they are received.

The RACF subsystem address space is a multitasking address space and can run many RRSF requests at the same time. To ensure that the commands issued by a user and the application updates initiated by a user run in the same order they are issued, RACF runs only one command or application update at a time for a given user. For example, if PAT and LAUREN send commands from NODEA to NODEB in the following order:

```
PAT     sends  command_1
LAUREN  sends  command_A
PAT     sends  command_2
PAT     sends  command_3
LAUREN  sends  command_B
```

they might be received on NODEB in the following order:

```
command_1
command_2
command_A
command_B
command_3
```

and executed in the following order:

```
command_1
command_A
command_2
command_B
command_3
```

The commands sent by PAT are received and run in the order that PAT sent them, and the commands sent by LAUREN are received and run in the order that LAUREN sent them, but command_B is sent after command_3 from NODEA, and runs before command_3 on NODEB. This is similar to what happens when an RRSF network is not configured and multiple TSO users issue RACF commands simultaneously. The order in which MVS chooses to service the TSO users determines the order in which the commands run.

RACLINK functions have a higher priority than directed commands or application updates so it is possible for a RACLINK command issued after a directed command

or application update to take effect before the directed command or application update runs. For example, if you direct a command to a user ID you have an association with, and then issue a RACLINK UNDEFINE to delete the association, the association might be deleted before the directed command runs, causing the directed command to fail.

Defining an RRSF environment

Table 6 summarizes the tasks involved in defining an RRSF environment. Most of these tasks are usually performed by a system programmer, some might be performed by a security administrator, and some might be performed by a system programmer and security administrator working together.

Table 6. Defining an RRSF environment—summary of tasks. Tasks that are not numbered can generally be done in any order.

Task	For more information, refer to...
STEP 1: Preparation	
Determine which systems will be part of the environment and how they will be related. • Decide on a unique RRSF logical node name for each node. • Find out the VTAM LU name for each node. • Decide which nodes will be single-system nodes, and which will be multisystem nodes. • Decide what mode each RRSF node will operate in, remote or local. • For each RRSF node that is to operate in remote mode, decide which RRSF nodes it will communicate with.	"The RRSF network" on page 128
Decide which RRSF functions you want to use on each RRSF node, and how you want to use them.	"Customizing a remote sharing environment" on page 174
Ensure that all systems to be in the environment are running: • RACF 2.2 or the RACF component of the z/OS or OS/390 SecureWay Security Server • MVS 4.3 or later • TSO/E 2.3 or later • VTAM 3.4 or later, or VTAM 3.3 with PTF UY59772	"System prerequisites" on page 143
Evaluate whether you require cryptographic teleprocessing support, and implement this if required.	"Ensuring the security of RACF database information in a network" on page 145
Ensure that the RACF template versions are compatible on all systems.	"RACF template version considerations" on page 143
Ensure that the RACF dynamic parse versions are compatible on all systems.	"RACF dynamic parse version considerations" on page 143
Ensure that the SETROPTS option settings are compatible on all systems.	"SETROPTS options considerations" on page 144
Ensure that installation exits are compatible on all systems.	"Installation exit considerations" on page 149
Ensure that password authentication algorithms are sufficient on all systems.	"Installation exit considerations" on page 149

Table 6. Defining an RRSF environment—summary of tasks (continued). Tasks that are not numbered can generally be done in any order.

Task	For more information, refer to...
Configure VTAM and APPC/MVS (for remote mode only) • Define NOSCHED LUs for RRSF nodes. • Specify VERIFY=REQUIRED on APPC LU definitions in SYS1.VTAMLST. • Create RACF profiles to protect APPC resources, specifying CONVSEC(ALREADYV) on the RDEFINEs. • Activate the APPCLU class, if not already activated. • Protect the ACBNAME used for RRSF. • Restrict access to the LU on the local system. • Define APPCPORT profiles to restrict access to LUs from remote systems. • Use APPCSERV profiles to protect APPC server access to the LU name associated with RRSF. • Activate the APPCPORT, APPCSERV, and APPCTP classes if not already active. • Control database token maintenance.	"VTAM and APPC/MVS considerations for an RRSF network" on page 145
Determine whether any of the systems in the environment have installation-provided code to update a remote database, and if so determine whether you need to remove the code.	"Considerations for installation-provided code" on page 151
Determine whether installation exits need to know which address space they've been given control in, and update them if necessary.	"Installation exit considerations" on page 149
If you are planning to have RACF maintain synchronization of any profiles between databases, synchronize those profiles.	"Synchronizing database profiles" on page 145
Create or modify the JCL to activate the RACF subsystem. Make sure the user ID for the RACF subsystem can access the RRSF resources.	"RACF subsystem address space considerations" on page 151
STEP 2: Configuration and customization	
Configure the RRSF network. On each node, create a RACF parameter library containing the configuration statements to configure the network from that node's point of view.	"Configuring an RRSF network" on page 152
Ensure that the RACF parameter library is protected, and that the user ID assigned to the RACF subsystem has authority to it.	"The RACF parameter library" on page 168
Ensure that the workspace data sets are protected, and that the user ID assigned to the RACF subsystem has authority to them.	The discussion of the WORKSPACE keyword in "Defining RRSF nodes to RACF" on page 156
Customize the RRSF environment by defining RRSFDATA profiles.	"Customizing a remote sharing environment" on page 174
Activate the RRSFDATA class on each RRSF node.	"Customizing a remote sharing environment" on page 174
STEP 3: Enabling RACF communications	
Restart the RACF subsystem on each RRSF node, to process the configuration statements in the node's RACF parameter library.	"RACF subsystem address space considerations" on page 151

Preparing to configure an RRSF network

This section identifies things to consider when preparing to configure an RRSF network.

System prerequisites

RRSF requires the following products on each node in the network:

- RACF 2.2 or the RACF component of the z/OS or OS/390 SecureWay Security Server
- MVS/ESA 4.3 or later
- TSO/E 2.3 or later

and for remote communications:

- APPC (part of MVS/ESA)
- VTAM 3.4 or later, or VTAM/ESA 3.3 with PTF UY59772

Nodes configured in local mode do not require APPC and VTAM.

When you use the TARGET command to establish communications between two RRSF nodes, RACF expects that both nodes are running RACF 2.2 or the z/OS or OS/390 SecureWay Security Server, and that both nodes are running APPC 4.3 or later.

Before you can define a multisystem node in an RRSF network, *each* system in the network must have one of the following installed:

- z/OS SecureWay Security Server Release 1 or later
- OS/390 Security Server (RACF) Release 2 or later
- RACF Version 2 Release 2 with APAR OW13567

To have application updates directed to or from a node, the node must have the Release 3 or higher OS/390 Security Server (RACF) installed.

RACF template version considerations

If systems in an RRSF network have different versions of the RACF templates installed, commands that run successfully on one system might fail on another one. When an RRSF node attempts to establish communications with another RRSF node, RACF determines which versions of the templates are installed on the two nodes and issues a warning message if they are different. You should ensure that all RRSF nodes that communicate with each other have the same version of the RACF templates installed. However, you can run with different template versions as long as you do not try to add or alter a profile using a field that exists in one system's templates but not in another's. Typically this would only be a concern with new fields in non-base segments, when automatic direction is active.

You can determine what level of RACF templates your system is using by issuing a RACF SET LIST operator command.

To update the RACF templates use the IRRMIN00 utility. For information on IRRMIN00 see "RACF database initialization utility program (IRRMIN00)" on page 208.

RACF dynamic parse version considerations

If systems in an RRSF network have different versions of the dynamic parse specification data set (IRRDPSDS) installed, commands that run successfully on one system might fail on another one. You can run with different dynamic parse versions as long as you do not try to add or alter a profile using a segment field that exists in one system's IRRDPSDS member but not in another.

When an RRSF node attempts to establish communications with another RRSF node, RACF determines which versions of the dynamic parse specification data set (IRRDPSDS) are installed on each node and issues a warning message if they are

different. You can determine what level of the dynamic parse specification data set your system is using by issuing a RACF SET LIST operator command.

To determine the contents of the dynamic parse table, issue the IRRDPI00 LIST command. See “Dynamic parse” on page 64 for more information about the dynamic parse table.

SETROPTS options considerations

If systems in an RRSF network have different SETROPTS options in effect, RACF commands or macros that run successfully on one system might fail on another one. Therefore, SETROPTS options should be compatible on systems that direct commands or application updates to each other, particularly if the systems use automatic direction. When an RRSF node attempts to establish communications with another RRSF node, RACF checks certain SETROPTS options in effect on each node and issues a warning message if any of the following options are not the same on both systems:

- EGN
- GENCMD(DATASET)
- GENERICOWNER
- PASSWORD(HISTORY(*x*))
- PASSWORD(INTERVAL(*x*))
- PASSWORD(RULE*x*)

For password rules, RACF checks whether the same rules are defined on each node. The rules do not have to be defined in the same order. For example, if two RRSF nodes each have two password rules, and the first rule is defined as RULE1 on the first node and as RULE2 on the second node, and the second rule is defined as RULE2 on the first node and as RULE1 on the second node, then RACF considers the password rules to be the same and does not issue a warning.

However, if a third rule is defined on the first node, but not on the second node, RACF detects a mismatch between the nodes. RACF then issues a warning for each RULE*x* that does not match on the two nodes. In this example, RACF would warn that RULE1 does not match on the two nodes, that RULE2 does not match on the two nodes, and that RULE3 does not match on the two nodes. Therefore, if one node contains a subset of the rules on another node, consider defining the common subset using the same SETROPTS PASSWORD(RULE*x*) commands, to reduce the number of warning messages RACF issues.

Evaluate the SETROPTS options in effect on each system, and for best results ensure that the SETROPTS options that RACF checks are the same on systems that communicate with each other. To change a SETROPTS option, use the SETROPTS command. For information on the SETROPTS command, see *z/OS Security Server RACF Command Language Reference*.

Class descriptor table considerations

If systems in an RRSF network have different versions of the class descriptor table (CDT) installed, commands and macros that run successfully on one system might fail on another one. For example, if a class is defined in the class descriptor table on one system, but is not defined on another system, or is defined differently, a command that specifies that class might run on the first system, but fail when directed to the other system.

For example, for OS/390 Version 2 Release 8, and for Version 2 Releases 6 and 7 with PTF UW90545 installed, the definitions of the SCICSTST and UCICSTST classes changed to allow longer profile names and longer resource names for

RACROUTE issuers that specify ENTITYX. If an RRSF network contains a system with z/OS Version 1 Release 1 installed and another system with OS/390 Version 2 Release 5 installed, a command that references an SCICSTST or UCICSTST profile or member with a longer name that runs on the z/OS system will fail if it is directed to the Release 5 system.

Ensuring the security of RACF database information in a network

RACF masks the data portion of RRSF message packets. The data is masked while the message packets are on the RRSF message queues and during transmission. This masking provides a default minimal level of confidentiality for the security-relevant information that these message packets carry. (This protection supplements the protection that classes such as APPCSERV and APPCLU provide to the data during transmission, and that RACF DATASET authorization provides to the data while it is in the workspace data sets.) The masking technique used for this purpose is the IBM Commercial Data Masking Facility (CDMF). The CDMF key has an effective strength of 40 DEA-key bits. RACF provides the CDMF algorithm and the key. There is no provision for changing the key.

RRSF data masking does not provide the protection that DES cryptography or even CDMF with installation-selectable keys could provide. The objective of RRSF data masking is to provide protection against inadvertent casual viewing of RACF profile data. The objective of RRSF data masking is *not* to provide confidentiality for RACF data, as might be provided if encryption with sophisticated key management were supported. It is strongly recommended that installations consider the use of cryptographic teleprocessing support in cases where RRSF is to be used across open (that is, non-secured) networks.

Synchronizing database profiles

You can use automatic direction to maintain synchronization of RACF database profiles that are already synchronized, but you must synchronize the profiles before you activate RRSF functions. You can do this synchronization manually, but it can be a time-consuming process. You can also run IRRDBU00 against the databases you want to synchronize, and use a program or REXX EXEC to compare the IRRDBU00 output for the databases and generate the commands needed to synchronize them. IBM provides a sample REXX EXEC, DBSYNC, to help you do this. IBM does not support the DBSYNC EXEC. For information on how to get this tool and others from the RACF home page or via anonymous FTP, see “Internet sources” on page xviii.

VTAM and APPC/MVS considerations for an RRSF network

APPC/MVS is the communications vehicle for sending and receiving messages from one RRSF node to another. You must configure VTAM for APPC/MVS and implement APPC/MVS before you can use RRSF in remote mode. This document assumes that you have a basic understanding of VTAM and APPC/MVS. For information on configuring VTAM and implementing APPC/MVS, see *z/OS MVS Initialization and Tuning Guide* and *z/OS MVS Planning: APPC/MVS Management*. See also *RACF Version 2 Release 2 Technical Presentation Guide* and *RACF Version 2 Release 2 Installation and Implementation Guide*.

When you define an RRSF node, you specify the LU name of the node on the TARGET command. The LU must be defined to VTAM on the node being TARGETed. LUs are defined through the LUADD statement in the APPCPM*xx* member of SYS1.PARMLIB. The LUs that you define for RRSF must be NOSCHED LUs.

We recommend that on the APPC LU (ACB) definitions in the SYS1.VTAMLST library concatenation you protect the information flowing between RRSF nodes by specifying VERIFY=REQUIRED.

If you specify VERIFY=REQUIRED, then in order to get proper RACF protection you must activate the APPCLU class and must code the parameter CONVSEC(ALREADYV) on the profiles in this class. The SETROPTS command issued to activate the APPCLU class should specify:

```
SETROPTS CLASSACT(APPCLU) +
        GENERIC(APPCLU)  +
        AUDIT(APPCLU)
```

You must create RACF profiles to protect the APPC resources. Assume a network with two nodes MVS1 and MVS2, for example. Node MVS1 needs a profile similar to the following for completion of the VERIFY=REQUIRED setup:

```
RDEFINE APPCLU netid.locallu.partnerlu UACC(NONE) +
     SESSION(SESSKEY(session-key) CONVSEC(ALREADYV))
```

or, if you have VTAM 4.1 installed with network-qualified names on (NQNAMES=YES):

```
RDEFINE APPCLU localnetid.locallu.partnernetid.partnerlu UACC(NONE) +
     SESSION(SESSKEY(session-key) CONVSEC(ALREADYV))
```

You get the *netid* or *localnetid* value in the RDEFINE command from the NETID keyword in the VTAM ATCSTR*xx* SYS1.VTAMLST member. The profile for node MVS1 might look like this:

```
RDEFINE APPCLU NET1.RM41MVS1.RM42MVS1 UACC(NONE) +
     SESSION(SESSKEY(session-key) CONVSEC(ALREADYV))
```

Node MVS2 needs a profile similar to the following to define the LU-LU relationship from its perspective:

```
RDEFINE APPCLU NET1.RM42MVS1.RM41MVS1 UACC(NONE) +
     SESSION(SESSKEY(session-key) CONVSEC(ALREADYV))
```

The SESSKEY value in the RDEFINE commands for MVS1 and MVS2 must be identical.

For RRSF you must specify CONVSEC(ALREADYV) on the RDEFINE for the APPCLU resources.

The RACF subsystem address space becomes an APPC/MVS server. It does this by registering through the Register_For_Allocates service of the APPC/MVS API. During the registration process, APPC/MVS uses RACF to determine if the caller is authorized to assume the server role for the requested transaction program. If the registration is successful, then APPC/MVS creates an allocate queue for the RRSF APPC server, which is a task within the RACF subsystem address space. The RRSF APPC server then becomes responsible for processing the allocate requests for which it has registered.

Protecting the ACBNAME used for RRSF: We recommend that you protect the ACBNAME used for RRSF. You can do this using the following definitions. You should first do a SETROPTS LIST and an RLIST VTAMAPPL * to see what you already have set up, so you do not repeat steps you have already done.

```
SETROPTS CLASSACT(VTAMAPPL)    +
         RACLIST(VTAMAPPL)     +  << Required
         AUDIT(VTAMAPPL)       +  << Optional
         GENERIC(VTAMAPPL)        << Optional, recommended
```

```
RDEFINE VTAMAPPL acbname UACC(NONE)

SETROPTS RACLIST(VTAMAPPL) REFRESH
```

You can use a generic profile to cover *acbname* instead of the discrete profile shown, but be careful if you do this because a generic profile could affect existing applications.

Controlling access to LUs on the local system: Consider whether you want to restrict access to the LU on the local system. RACF requests coming from remote systems as well as other APPC/MVS traffic are received on this LU. Therefore, you might want to only grant access to those users who need access to this information, such as the local RACF subsystem user ID. You can use the following RACF definitions to control access to the LU:

```
SETROPTS CLASSACT(APPL)  +  << Required
         AUDIT(APPL)     +  << Optional
         RACLIST(APPL)   +  << Optional, recommended for
                               performance reasons
         GENERIC(APPL)   +  << Optional, recommended

RDEFINE APPL luname UACC(NONE) NOTIFY(administrator)
PERMIT luname CLASS(APPL) ID(userid) ACCESS(READ)
```

The *userid* value is the user ID (or associated group name) that the local RACF subsystem is operating under. This definition basically restricts the usage of the LU name to the RACF subsystem.

Controlling access to LUs from remote systems: To control which remote users or applications can access the local RRSF system, define APPCPORT profiles with the names of the remote LUs, and selectively give READ access to the user ID associated with the remote RACF subsystems. This is an optional step which you might or might not choose to do depending on whether you are protecting access to your LUs today. For example:

```
RDEFINE APPCPORT partner-luname UACC(NONE)

PERMIT partner-luname CLASS(APPCPORT) ID(userid or group)
       ACCESS(READ)

SETROPTS CLASSACT(APPCPORT) RACLIST(APPCPORT)
```

The *userid or group* parameter specifies the user ID associated with the incoming request. The *partner-luname* parameter specifies the locally known name of the partner LU. If the APPC LUADD statement for the LU specifies the NQN option, the partner LU name is a network-qualified name of 1 to 17 characters in the form *netid.luname.* If the APPC LUADD statement does not specify the NQN option, the partner LU name is an unqualified LU name of 1 to 8 characters. Any time an APPCPORT profile is changed, SETROPTS RACLIST processing for the APPCPORT class must be refreshed in order for the change to take effect.

APPC TP profiles: RACF does not need a TP profile within a TP profile data set. But in order for the RACF subsystem address space to register as an APPC server (see “Providing security for server access to specific LU or TP names” on page 148), a DBTOKEN must be associated with a TP profile data set. You can use a DBTOKEN associated with a TP profile data set that is already in use for APPC/MVS, such as the one being used by the APPC/MVS scheduler (ASCH). If you need to create a TP profile data set and get a DBTOKEN, see *z/OS MVS Planning: APPC/MVS*

Management for additional information. No APPC side information profile is needed for RACF, and no profiles are required in the TP profile data set for RACF.

Providing security for server access to specific LU or TP names: You should use APPCSERV profiles to protect APPC server access to the LU name associated with RRSF. The APPC/MVS server facilities perform security verification when the RACF subsystem address space attempts to register as an APPC/MVS server. APPC/MVS checks the access of the user ID assigned to the RACF subsystem address space to a profile defined to RACF in the APPCSERV general resource class. The profile for this checking has the following format:

```
dbtoken.tpname
```

where

dbtoken
: Is the database token (1 to 8 characters) of the TP profile data set. The TP profile data set is associated with the LU at which the server resides. (This is the LU that the RRSF APPC server specifies on the *local-luname* parameter of the Register_For_Allocates service.)

tpname
: Is the name of the transaction program (1 to 64 characters) to be served. Unless the installation changes it, RACF uses the default TPNAME of IRRRACF.

To register for a particular TP name, the user ID under which the server runs (the user ID assigned to the RACF subsystem) must have been granted READ access to the TP's security profile in the APPCSERV RACF general resource class. For example:

```
RDEFINE APPCSERV dbtoken.tpname UACC(NONE)
PERMIT dbtoken.tpname CLASS(APPCSERV)
       ID(subsystem-userid) ACCESS(READ)
SETROPTS CLASSACT(APPCSERV)
```

If the TP name is not protected by the APPCSERV class, and the APPCSERV class is active, APPC/MVS fails the registration request.

Controlling access to the transaction program profiles: Inbound requests for a local RACF subsystem are handled by a program which is invoked by an APPC transaction program profile process. This profile must be protected in order to prevent undesirable alterations which could bypass security processes, and to control which remote users can send inbound requests.

There are two steps in controlling a transaction program profile:

1. Protect the VSAM data set containing the profile. The level of protection should restrict who can alter the profile. You might also want to restrict who can read the data set. In this case, we recommend that the ERASE attribute be specified on the DEFINE for the VSAM cluster.
2. Protect the associated transaction program profile from unauthorized execution of inbound requests.

Both of these steps can be performed through the use of the APPCTP class. Profiles in this class have the form:

```
dbtoken.tplevel.tpname
```

where

dbtoken
Is the database token (1 to 8 characters) for the TP profile data set.

tplevel Is the transaction program level. This *tplevel* corresponds with the TPLEVEL specified on the LUADD. For example, if you specify TPLEVEL(USER) on the LUADD, APPC looks for an APPCTP profile protecting *dbtoken.userid.tpname*. There is no RACF requirement for the TPLEVEL. See the APPC manuals referenced in "VTAM and APPC/MVS considerations for an RRSF network" on page 145 for information.

tpname
Is the transaction program name (1 to 64 characters). Unless the installation changes it, RACF uses the default TPNAME of IRRRACF.

The local RACF user ID authorized to this profile must be the same as the user ID that the RACF subsystem in the remote node operates under.

Controlling database token maintenance: The profiles in the APPCTP class make use of database token values. These values are maintained with the DBRETRIEVE and DBMODIFY commands of the APPC administration utility. The profile for protecting database tokens is defined to the RACF FACILITY class. The profile is of the form:

```
APPCMVS.DBTOKEN
```

We suggest that you define this profile as follows:

```
RDEFINE FACILITY APPCMVS.DBTOKEN UACC(NONE)
```

You should then PERMIT the proper user IDs and groups to this profile with the appropriate access authority:

NONE Not allowed to retrieve or modify the database tokens.

READ Allowed to perform DBRETRIEVE on existing database tokens.

UPDATE
Allowed to perform DBRETRIEVE and DBMODIFY for the installation.

Each TP profile data set should have a database token defined for it. APPC/MVS does not check access requests if there is no database token defined for the TP profile data set.

Installation exit considerations

When a command is directed to another node it uses the installation exits and naming convention table of the node on which it runs, *not* of the node on which it is issued. If you plan to use RRSF in remote mode, you should first ensure that any installation exits and naming convention tables that you have on the RRSF nodes are compatible. Otherwise, a command might have different results when it runs on different nodes.

If you plan to use password synchronization or automatic password direction you are not required to use the same password authentication algorithm on each RRSF node, but we recommend that you do so. Although nodes can have different ICHDEX01 exits and still synchronize passwords, if you have security reasons for using a stronger algorithm on one node, it is advisable to use the stronger algorithm on all nodes that synchronize passwords. A system with a stronger algorithm is as vulnerable as one using a weaker algorithm if they synchronize passwords, because a password that is compromised on the weaker system can then be used on the stronger system.

Directed commands and application updates run in the RACF subsystem address space rather than a user's address space. Installation exits that need to know whether they have been given control in the RACF subsystem address space or a user's address space should check the ACEERASP bit in the ACEE. The ACEERASP bit is on in the address space level ACEE of the RACF address space.

When automatic direction of application updates is active, RACROUTE REQUEST=DEFINE exits can instruct RACF to propagate installation data that is passed to them. For more information, see "Automatic direction of application updates" on page 284.

During the execution of a RACF command or macro, one or more installation exits might be invoked. If one of these installation exits issues a RACF command, that command is referred to as an *exit-generated command.* If one of these installation exits issues a RACROUTE macro or ICHEINTY macro to update the RACF database, that macro is referred to as an *exited-generated macro.* If automatic direction is in effect, the exit-generated command or macro might or might not be propagated by automatic direction, according to the following rules:

- If the original RACF command or macro is not eligible for direction, but the exit-generated command or macro is eligible for direction, RACF propagates the exit-generated command or macro based on the RRSFDATA profiles. For example, if an application issues a RACROUTE REQUEST=AUTH macro, which is not eligible for direction, and the ICHRCX01 exit issues a RACROUTE REQUEST=DEFINE, which is eligible for direction, RACF propagates the RACROUTE REQUEST=DEFINE if the RRSFDATA profiles specify that it should be directed.
- If the original RACF command or macro is eligible for direction, the exit-generated command or macro might or might not be propagated:
  - If the exit-generated command or macro runs under the same task (TCB) as the original RACF command or exit, automatic direction does not propagate the exit-generated command or macro. For example, if an exit invokes a RACF command using the LINK macro, the command is not propagated. However, as long as you have the same exit on two systems, the exit issues the command or macro once on each system, and the RACF databases should remain synchronized.
  - If the exit-generated command or macro runs under a different task (TCB) than the task for the original RACF command or exit (for example, an exit creates a new task which runs a command) then the exit-generated command or macro is propagated based on the RRSFDATA profiles. For example, if an exit invokes a RACF command using the ATTACH macro, the command is propagated if the RRSFDATA profiles specify that it should be directed.

    In this situation, if two or more RRSF nodes have the same installation exit, the exit-generated command or update might be propagated more than once and produce unpredictable results.

    Example: Assume that the RACROUTE REQUEST=DEFINE postprocessing exit (ICHRDX02) called from the `ADDSD 'JOE.*'` command creates a subtask using the ATTACH macro to run the command `RDEFINE FACILITY JOE.*`. Assume that the same ICHRDX02 exit is present on both NODE1 and NODE2, and that automatic command direction has been activated to propagate all DATASET and FACILITY class commands between NODE1 and NODE2.

    When `ADDSD 'JOE.*'` is running on NODE1, the exit issues the RDEFINE command, and automatic command direction propagates that command to NODE2. Meanwhile, on NODE2 the propagated `RDEFINE FACILITY JOE.*`

command runs and completes successfully. On NODE1 the ADDSD command completes successfully and automatic command direction propagates the command to NODE2. Next, on NODE2 the propagated `ADDSD 'JOE.*'` begins. The exit on NODE2 gets control and ATTACHes the RDEFINE command on NODE2, which fails with return code 4 because it already ran once on NODE2.

Automatic command direction then propagates the RDEFINE command to NODE1 where it also fails with return code 4. Therefore, the RDEFINE command runs twice on each node. The second RDEFINE command on each node is unnecessary and fails.

Although this example does not cause the RACF databases to become unsynchronized, your installation exits might issue RACF commands or macros that can cause synchronization problems.

Considerations for installation-provided code

If any of the systems in your RRSF network have installation-provided code on them that updates a remote database, you might have to remove that code. You need to carefully evaluate what RRSF functions you plan to use and what functions the installation-provided code performs to determine whether to remove the code. If the code performs a function that you plan to use RRSF to perform, then you should remove the code to avoid duplicate updates to the RACF database.

For example, if you plan to use RRSF to synchronize passwords, and if you have installation-provided code that synchronizes passwords, remove the installation-provided code from all systems in the RRSF network before you activate RRSF password synchronization. However, if you are not planning to use RRSF password synchronization, you can continue to use your installation-provided code to synchronize passwords.

Similarly, if you plan to synchronize databases via automatic command direction, automatic password direction, and automatic direction of application updates, but you already have installation-provided code that synchronizes a remote database, you should probably remove that code. It is up to the installation to choose which RRSF options are turned on and which installation-provided code is removed due to redundancy.

RACF subsystem address space considerations

The RACF remote sharing facility requires that the RACF subsystem address space be active. If you want to use remote sharing functions on an MVS system image, you must first activate the RACF subsystem address space. See “Activating the RACF subsystem” on page 70 for information on how to do this.

If you already run with the RACF subsystem address space active, you must update your JCL if you want RACF to automatically process a member of the RACF parameter library when the RACF subsystem address space is activated. See “The RACF parameter library” on page 168 for information on the RACF parameter library. See “The RACF PROC” on page 75 for information on how to update your JCL.

You must ensure that the user ID assigned to the RACF subsystem has access to the resources that remote sharing functions use, including:

- Workspace data sets
- APPC-related profiles
- The RACF parameter library

We recommend that you make this user ID trusted or privileged. See “Assigning a user ID to the RACF subsystem” on page 74.

Selecting the main system for a multisystem node

The main system in a multisystem node receives the majority of RRSF traffic to the node. Be sure to select a system that can handle the RRSF traffic. To minimize the time that RRSF requests will have to be kept in workspace data sets while the system is unavailable, the main system should also be the system least likely to need hardware or software changes.

When deciding the file size for the main system, consider the volume of RRSF traffic the system will handle, and the amount of time it might be unavailable.

Configuring an RRSF network

Before an RRSF node can communicate with other RRSF nodes, you must provide the node with information about its own operational characteristics and the operational characteristics of the nodes with which it is to communicate. The node you are configuring is referred to as the *local node*. The nodes with which it is to communicate, not including itself, are referred to as *remote nodes*. All of the nodes with which the local node is to communicate, including itself and its remote nodes, are referred to as *target nodes*. RACF provides the following RRSF configuration facilities:

- The SET command, to define operational characteristics of the local system
- THE TARGET command, to define operational characteristics related to communications for a node
- An optional RACF parameter library, which allows you to specify standard predefined sequences of configuration commands. These sequences can be automatically processed during initialization of RRSF, or manually processed by entering RACF operator commands.

To configure an RRSF network, you must configure each node in the network. To configure a node, run RACF configuration commands (SET and TARGET) on the node. The node you are configuring is the local node for the configuration commands that run on it. For example, suppose you have a network with two nodes, NODEA and NODEB. Then, to configure the network, you must run configuration commands on NODEA defining the characteristics of NODEA, the local node, and NODEB, a remote node for NODEA. You must also run configuration commands on NODEB defining the characteristics of NODEB, the local node, and NODEA, a remote node for NODEB. See “Configuring a two-node network” on page 178 for an example of commands you could use to configure this two-node network.

The SET command

The SET command specifies the operational characteristics of the local system. This command allows you to:

- List the attributes of the local node. See “Listing the attributes of the local node” on page 154.
- Specify trace parameters. See “Tracing APPC and IMAGE events” on page 154.
- Specify one or more members of the RACF parameter library to process. For more information on the RACF parameter library, see “The RACF parameter library” on page 168.
- Specify whether password synchronization is active.
- Specify whether commands, password changes, or application updates are automatically directed from the local node.

- Specify whether any user IDs will receive output from automatically directed commands, automatically directed passwords, synchronized passwords, or automatically directed application updates executed at the local node, and if so which user IDs and under what conditions.
- Specify whether any user IDs will be notified whether automatically directed commands, automatically directed passwords, synchronized passwords, or automatically directed application updates executed at the local node succeed or fail, and if so which user IDs and under what conditions.

The syntax of the SET command is shown in Figure 14.

```
subsystem-prefixSET
                    [ AUTOAPPL[(
                      [ NOTIFY(notify-level(list-of-notify-users))
                        | NONOTIFY ]
                      [OUTPUT(output-level(list-of-output-users))
                        | NOOUTPUT ]
                      )]
                      | NOAUTOAPPL
                      ]
                    [ AUTODIRECT[(
                      [ NOTIFY(notify-level(list-of-notify-users))
                        | NONOTIFY ]
                      [OUTPUT(output-level(list-of-output-users))
                        | NOOUTPUT ]
                      )]
                      | NOAUTODIRECT
                      ]
                    [ AUTOPWD[(
                      [ NOTIFY(notify-level(list-of-notify-users))
                        | NONOTIFY ]
                      [OUTPUT(output-level(list-of-output-users))
                        | NOOUTPUT ]
                      )]
                      | NOAUTOPWD
                      ]
                    [ INCLUDE(member-suffix...) ]
                    [ JESNODE(nodename) ]
                    [ LIST ]
                    [ PWSYNC[(
                      [ NOTIFY(notify-level(list-of-notify-users))
                        | NONOTIFY ]
                      [OUTPUT(output-level(list-of-output-users))
                        | NOOUTPUT ]
                      )]
                      | NOPWSYNC
                      ]
                    [ TRACE( {[ APPC | NOAPPC ] [ IMAGE | NOIMAGE ] } ) ]
```

Figure 14. SET command syntax

The SET command can be issued as a RACF operator command, or from the RACF parameter library. For further information on the syntax of the SET command, see *z/OS Security Server RACF Command Language Reference*. For information on establishing security for the SET command, see *z/OS Security Server RACF Security Administrator's Guide*.

Listing the attributes of the local node

Use the **LIST** keyword on the SET command to list the attributes of the local node. The LIST keyword is the default if no other keywords are specified. Figure 15 illustrates the type of information displayed.

```
NODE1  IRRH005I (<) RSFJ SUBSYSTEM INFORMATION:
   TRACE OPTIONS                   - NOIMAGE
                                   - APPC
   SUBSYSTEM USERID                - IBMUSER
   JESNODE (FOR TRANSMITS)         - POKVMMCL
   AUTOMATIC COMMAND DIRECTION IS *NOT* ALLOWED
   AUTOMATIC PASSWORD DIRECTION IS *NOT* ALLOWED
   PASSWORD SYNCHRONIZATION IS *NOT* ALLOWED
   AUTOMATIC DIRECTION OF APPLICATION UPDATES IS *NOT* ALLOWED
   RACF STATUS INFORMATION:
         TEMPLATE VERSION          - HRF7708 00000020.00000010
         DYNAMIC PARSE VERSION     - HRF7708
```

Figure 15. Sample output from the SET LIST command. The value shown for the template version and dynamic parse version is the last APAR that affected the part, or an FMID such as HRF7708 if it has had no APAR service. The template version also includes an 8-digit field representing the release level and an 8-digit field representing the APAR level.

Tracing APPC and IMAGE events

You can obtain trace records for IMAGE or APPC events using the Generalized Trace Facility (GTF). The trace record type is EF44. The trace records are intended for use in consultation with the IBM support center when diagnosing possible RACF subsystem problems.

If the IBM support center requests trace records for IMAGE or APPC events, use the **TRACE** keyword on the SET command to set the trace parameters. Note that trace records can contain passwords, so be sure that trace output data sets are appropriately protected.

For example, to trace APPC events but not IMAGE events, enter:

```
SET TRACE(APPC NOIMAGE)
```

To continue tracing APPC events, and to turn on IMAGE tracing, enter:

```
SET TRACE(IMAGE)
```

Because no setting for APPC events is specified, the current setting remains in effect. To turn off both APPC and IMAGE tracing, enter:

```
SET TRACE(NOIMAGE NOAPPC)
```

Activating and deactivating RRSF functions

The RACF SET command activates and deactivates RRSF functions on the local node. Use the **AUTODIRECT** keyword to activate automatic direction of commands. Use the **AUTOPWD** keyword to activate automatic direction of passwords. Use the **PWSYNC** keyword to activate password synchronization. Use the **AUTOAPPL** keyword to activate automatic direction of application updates. Before you issue the SET command to activate these functions, define profiles in the RRSFDATA class to control:

- Which commands or application updates are automatically directed and to which remote nodes
- Which users' password changes are to be synchronized
- Which users' password changes are automatically directed and to which remote nodes

If the RRSFDATA class is not active, no password changes are synchronized with other user IDs, even if password synchronization was activated with the SET PWSYNC command. Similarly, if the RRSFDATA class is not active, no commands, application updates, or password changes are automatically directed even if the function was activated via the SET command.

Use the **OUTPUT** and **NOOUTPUT** subkeywords on the AUTODIRECT, AUTOPWD, PWSYNC, or AUTOAPPL keywords to specify whether output, warning messages, and error messages from RRSF functions are sent to anyone. If you specify OUTPUT, you can specify a list of users to whom the output is sent and under what conditions. RACF puts the output in the RRSFLIST data sets of the users.

Use the **NOTIFY** and **NONOTIFY** subkeywords on the AUTODIRECT, AUTOPWD, PWSYNC, or AUTOAPPL keywords to specify whether RACF should issue TSO SEND commands indicating whether the RRSF functions were successful or unsuccessful. If you specify NOTIFY, you can specify a list of users to whom the notification is sent and under what conditions.

The SET command also deactivates RRSF functions. Use the **NOAUTODIRECT** keyword to deactivate automatic direction of commands. Use the **NOAUTOPWD** keyword to deactivate automatic direction of passwords. Use the **NOPWSYNC** keyword to deactivate password synchronization. Use the **NOAUTOAPPL** keyword to deactivate automatic direction of application updates.

For more information on the SET command and controlling RRSF functions, see *z/OS Security Server RACF Security Administrator's Guide*.

Specifying a parameter library member to process

You can use the **INCLUDE** keyword on the SET command to specify a parameter library member to process. See "Using the SET INCLUDE function" on page 171.

Specifying a JES node to return output to

Use the **JESNODE** keyword on the SET command to specify a JES node to return output to. When RACF is unable to put output from a directed command or application update in the user's RRSFLIST data set, RACF transmits the output to the user. During initialization RACF queries the primary JES in order to obtain the JES node name. JES releases earlier than JES2 4.3 and JES3 5.1.1 do not support the query of the node name. If you are using one of these releases, specify the JESNODE keyword to provide the JES node name to RACF. If you specify the JESNODE keyword and you are using a JES release that does support the query of the node name, the node name you specify overrides the node name obtained from the query.

The TARGET command

The TARGET command specifies the operational characteristics of a node with which the local node is to communicate. Each node that the local node expects to communicate with must be defined by a TARGET command, including the local node itself. These nodes are referred to as *target nodes*. The TARGET command allows you to:

- List the attributes of specified nodes. See "Listing the attributes of target nodes" on page 161.
- Specify whether you are defining the local node or a remote node. See "Defining RRSF nodes to RACF" on page 156.
- Specify whether a node is a single-system node or a multisystem node. See "Defining RRSF nodes to RACF" on page 156.

- For a multisystem node, identify the systems that make up the node, and specify which system is the main system. See “Defining RRSF nodes to RACF.”
- Specify the name of a node. See “Defining RRSF nodes to RACF.”
- Describe a node. See “Defining RRSF nodes to RACF.”
- Specify whether the connection to the target node is dormant or operative. See “Controlling outgoing requests from the local node” on page 164.
- Delete an RRSF node from the set of target nodes known to the local node. See “Deleting a node” on page 165.
- Purge the workspace data sets for an RRSF node. See “Purging a workspace data set” on page 165.
- Specify information about the network transport mechanism, APPC. See “Defining RRSF nodes to RACF.”
- Specify a prefix for the workspace data sets that RACF allocates for each target node. See “Defining RRSF nodes to RACF.”
- Specify the characteristics for the workspace data sets that RACF allocates for each target node. See “Defining RRSF nodes to RACF.”

The syntax of the TARGET command is shown in Figure 16.

```
subsystem-
prefixTARGET
                    [ DELETE | DORMANT | OPERATIVE ]
                    [ DESCRIPTION('description') ]
                    [ LIST ]
                    [ LOCAL ]
                    [ MAIN ]
                    [ NODE(nodename|*) ]
                    [ PREFIX(qualifier...) ]
                    [  PROTOCOL(APPC(LUNAME(luname)
                       [  TPNAME(profile-name)  ]
                       [  MODENAME(mode-name)  ]
                       ))]
                    [ PURGE(INMSG | OUTMSG) ]
                    [ SYSNAME(sysname | *) ]
                    [ WDSQUAL(qualifier) ]
                    [  WORKSPACE(
                       { [  STORCLASS(classname)  ]
                             [  DATACLAS(classname)  ]
                             [  MGMTCLAS(classname)  ]
                       | [  VOLUME(volume-serial)  ] }
                       [  FILESIZE([nnnnnnnnnn|500])  ]
                       )]
```

Figure 16. TARGET command syntax

The TARGET command can be issued as a RACF operator command, or from the RACF parameter library. For further information on the syntax of the TARGET command, see *z/OS Security Server RACF Command Language Reference*. For information on establishing security for the TARGET command, see *z/OS Security Server RACF Security Administrator's Guide*.

Defining RRSF nodes to RACF

Use the TARGET command to define RRSF nodes to RACF. You must specify a node name and workspace information for a node before you can make the connection with the node operative. In addition, if the node is not running in local

mode, you must also specify protocol information before you can make the connection with the node operative. You can also specify a description of the node, and the high level qualifiers to be used on the INMSG and OUTMSG data sets for the node.

An MVS system image must meet the following requirements to be defined as an RRSF node:
- MVS/ESA 4.3 or later is installed.
- RACF 2.2 or the RACF component of the z/OS or OS/390 SecureWay Security Server is installed.
- The RACF subsystem address space is active.

You must define the local node and the local node must be in a dormant or operative state before you can make any other nodes operative, because RACF uses the local LU name to allocate the workspace data sets for the target nodes. In addition, if the local node is a multisystem node you must define the main system of the remote node before you can make any system in that remote node operative.

RACF cannot determine whether the member systems of a multisystem node share a RACF database. If you configure a multisystem node, you must ensure that its member systems share a RACF database.

RACF checks the TARGET commands issued for a node by that node and by other nodes, and can detect mismatches in:
- LU names
- Node names
- System names
- The system designated to be the main system
- A node's definition as a multisystem node or a single-system RRSF node

For example, assume NODEA, NODEB, and NODEC have LU names LUA, LUB, and LUC respectively, and that NODEB and NODEC issue correct TARGET commands. Assume that NODEA incorrectly specifies LUC for NODEB and LUB for NODEC on its TARGET commands. When NODEA issues a TARGET OPERATIVE command for NODEC, RACF determines that there is a mismatch between the LU name specified for NODEC on NODEA and the LU name specified for NODEC on NODEC. RACF sets the connection to the operative pending verification state, and issues message IRRI014I. The message contains a reason code to help diagnose the mismatch so that the TARGET commands can be redone.

Use the **NODE** keyword on the TARGET command to specify a name for a node. This is the name that users will specify on the RACLINK command and the AT and ONLYAT keywords to identify the node. Therefore, the name should be something that is meaningful to users. The node name must meet the following restrictions:
- 1 to 8 characters in length
- First character is either A-Z, #, $, or @ (where # = X'7B', $ = X'5B' and @ = X'7C')
- Second through eighth characters are A-Z, 0-9, #, $, or @ (where # = X'7B', $ = X'5B' and @ = X'7C')

Use the **SYSNAME** keyword with the NODE keyword to identify which system on a multisystem node the command pertains to. If you specify the SYSNAME keyword, you must also specify the NODE keyword. The SYSNAME keyword is required on TARGET commands for multisystem nodes, unless the LIST keyword is specified or defaulted to. (See "Listing the attributes of target nodes" on page 161 for

information on when the LIST keyword is defaulted to.) If SYSNAME is not specified, and LIST is not specified or used as the defaulted, RACF assumes that the node is a single-system node.

sysname

- Is 1 to 8 characters long
- Can contain the characters A-Z, 0-9, $, @, and #
- Should not have a numeric as the first character because it is used as a data set qualifier for the local workspace data sets.

  Note: If *sysname* begins with a numeric that cannot be changed, specify WDSQUAL on the TARGET command to provide a replacement value for the data set qualifier in the local workspace data set name.
- Must match the value in the CVTSNAME field of the system the TARGET command describes. This is the SYSNAME specified in the IEASYS*xx* member of SYS1.PARMLIB.

You can specify an asterisk on the SYSNAME keyword to indicate that the command should be executed for each system in the multisystem node specified by the NODE keyword. For example, if you specify SYSNAME(*) with the LIST keyword and a NODE keyword of NODE(HURLEY), RACF generates a list for each system in the multisystem node named HURLEY. You can specify an asterisk on the SYSNAME keyword only in combination with the following keywords:

- NODE
- DORMANT
- OPERATIVE
- DELETE
- PURGE
- LIST

If LIST is specified with NODE(*), SYSNAME must be specified as SYSNAME(*) or omitted.

The SYSNAME keyword allows you to use a common set of TARGET commands on all the systems in a multisystem node. When the TARGET command is for a local node, and the OPERATIVE or DORMANT keyword is specified, RACF compares the SYSNAME specified on the TARGET command with the CVTSNAME for the system the command is to run on. If they do not match, RACF does not process the OPERATIVE or DORMANT keyword. In addition, because a conversation should not exist between the systems of a multisystem node, RACF issues an informational message and places it in the SYSLOG. This message might help diagnose why an expected conversation was not established.

Use the **MAIN** keyword to identify the system named on the SYSNAME keyword as the main system in a multisystem RRSF node. (For information about the main system, see “Single-system nodes and multisystem nodes” on page 129.) You must execute a TARGET command identifying the main system for a multisystem node on each system in the multisystem node, and on each RRSF node that communicates with the multisystem node. You must designate the same system as the main system on the local node and all other nodes that communicate with it. You must identify the main system of a multisystem node before you make any systems in the multisystem node operative.

For information on choosing the main system, see “Selecting the main system for a multisystem node” on page 152.

Use the **LOCAL** keyword on the TARGET command to identify the node you are defining as the local node. If you do not specify LOCAL, RACF assumes that the node is a remote node. You can only define one local node. Once you have defined a node as the local node, you do not have to specify LOCAL on subsequent TARGET commands for that node. If you plan to run in local mode, the only TARGET command you need is one for the local node.

Use the **DESCRIPTION** keyword on the TARGET command to specify a description of the node you are defining. The description will be displayed in the TARGET LIST output for the node.

Use the **PROTOCOL** keyword to indicate that the node being defined is to run in remote mode. The subkeywords of PROTOCOL specify information about the network transport mechanism, APPC. Protocol information is required in order to communicate with remote nodes. Protocol information should not be specified for a node running in local mode. If it is specified, the local node requires APPC to become operative, and unnecessary processing occurs.

Use the **LUNAME** subkeyword on the PROTOCOL keyword to identify the logical unit to be associated with the RRSF node being defined. Specify a 1-to-8-character LU name, or a qualified LU name in the form *netid.luname*, where *netid* is a 1-to-8-character network name, and *luname* is a 1-to-8-character LU name. You can find the LU name in the SYS1.PARMLIB APPCPM*xx* member on the target node you are defining. (The name specified on the ACBNAME keyword is the LU name.) You can also use the DISPLAY APPC operator command on a node to display its LU name. See *z/OS MVS Planning: APPC/MVS Management* for information on the DISPLAY APPC command. Some points to keep in mind:

- You must specify the LU name for a node before you can make the connection with that node operative.
- You can modify a node's LU name only while the node is in the initial state.
- If you specify the same LU name on multiple TARGET commands, the first usage takes precedence. For example, if you issue two TARGET commands for different node names, but with the same LU names, the node specified on the first TARGET command is associated with the LU name, and a message is issued when the second TARGET command is issued.
- On the local system, you must specify an LU name in the target definitions for the local peer members, even though conversations do not occur with these members. And the LU name specified for a particular remote target definition must match the LU name specified on all the local peer systems for their corresponding remote target definitions. If you later reconfigure the multisystem node with a new main system, the old main system's workspace data sets will be accessed by the new main system. If the LU names are not the same, the reconfiguration will not work.
- If the LU name is a qualified name in the form *netid.luname*, RACF uses only the second part of the qualified name as a qualifier for the workspace data set names. If the second part of the qualified LU name is not unique within the group of DASD devices shared by the local system, you must use the WDSQUAL keyword to supply a unique qualifier for the workspace data set names.

Use the optional **TPNAME** subkeyword on the PROTOCOL keyword to identify the APPC transaction program (TP) profile. The TP profile name is 1 to 64 characters in length, and defaults to IRRRACF. We recommend that you let RACF take the default and use IRRRACF as the TPNAME unless you need to change it. Once you have specified a TP profile name for a node, you can change it only if you first make the node dormant.

Use the optional **MODENAME** subkeyword on the PROTOCOL keyword to specify the mode name which designates the network properties for the session to be allocated. The mode name is an alphanumeric string 1 to 8 characters in length. If you do not specify a mode name, the TARGET LIST output shows the mode name as <NOT SPECIFIED>, and RACF uses the default name IRRMODE. VTAM issues an error message, and uses the default values for the session. If you want to prevent the error message from VTAM, use the sample VTAM LOGMODE entry for IRRMODE provided in member IRRSRRSF in SYS1.SAMPLIB. Once you have specified the APPC mode name for a node, you can change it only if you first make the node dormant.

Use the **PREFIX** keyword to specify one or more high level qualifiers for the data set names of the INMSG and OUTMSG queues. The maximum length of the prefix is 19 characters, and it can contain multiple qualifiers separated by periods. Your prefix should not end in a period, as RACF appends one to the end for you. Keep in mind when defining your prefix that, as for all data sets, the high level qualifier of the workspace data sets must be a RACF-defined user ID or group name.

We recommend that you specify the same prefix for all systems on a multisystem node, to allow you to reconfigure the multisystem node with a different main system. On the local system the prefix specified for a particular remote target definition must match the prefix specified on all the local peer systems for their corresponding remote target definitions. For example, if system MVSB of local node NODEAB defines the prefix for remote node NODEZ to be SYSZ, then system MVSA of local node NODEAB must also define the prefix for remote node NODEZ to be SYSZ. If you later reconfigure the multisystem node with a new main system, the old main system's workspace data sets will be accessed by the new main system. If the prefixes are not the same, the reconfiguration will not work.

Use the **WDSQUAL** keyword to specify a qualifier for a workspace data set name that RACF is to use instead of the value it uses by default.

- For a local node, use the WDSQUAL keyword to provide an alternative to the name specified on the SYSNAME keyword. If the system name specified on SYSNAME begins with a numeric character, you *must* specify WDSQUAL to provide a qualifier for the workspace data set names that does not begin with a numeric.
- For a remote node, use the WDSQUAL keyword to provide an alternative to the name specified on the LUNAME keyword. If the LU name is a qualified name, and the second part of the name would not be a unique data set qualifier within the group of DASD devices shared by the local system, you *must* use the WDSQUAL keyword to provide a unique qualifier for the workspace data set names.

For more information on the workspace data sets, see "Workspace data sets" on page 134.

Use the **WORKSPACE** keyword to specify the attributes of the workspace data sets for the INMSG and OUTMSG queues. There is an INMSG and OUTMSG workspace data set for each target node. You can preallocate the VSAM files for the workspace data sets yourself, or let RACF allocate them. If you preallocate the VSAM files, you do not need to specify the WORKSPACE keyword. For more information on the INMSG and OUTMSG workspace data sets, see "Workspace data sets" on page 134.

We recommend that you protect the workspace data sets you define from viewing, reading, and writing by unauthorized users. The user ID assigned to the RACF subsystem must have authority to allocate and write to these data sets.

You can specify either a System Managed Storage (SMS) or non-SMS workspace. It is very important that the workspace data sets do not run out of space. For this reason, we recommend that you create them as SMS-managed data sets that can grow as needed.

For multisystem nodes, we recommend that you allocate the workspace data sets on shared DASD using shared catalogs. Doing this allows you to share one set of TARGET definitions for all systems in a multisystem node.

- Use the **STORCLAS**, **DATACLAS**, and **MGMTCLAS** subkeywords on the WORKSPACE keyword to specify an SMS workspace. You must specify STORCLAS for an SMS workspace if you have not preallocated the VSAM files; DATACLAS and MGMTCLAS are optional.
- Use the **VOLUME** subkeyword on the WORKSPACE keyword to specify a non-SMS workspace. The volume serial number specified must be a valid volume on the system where the TARGET command is issued.

Use the optional **FILESIZE** subkeyword on the WORKSPACE keyword to specify how much space to allocate for the workspace data sets. Enough space is allocated for each data set to contain the number of entries specified on the FILESIZE subkeyword. Specify a number in the range 1 to 2 147 483 647. The initial value is FILESIZE(500). For some guidelines on what size to specify, see “Size guidelines for the workspace data sets” on page 136.

Listing the attributes of target nodes

Use the **LIST** keyword on the TARGET command to list the attributes of RRSF nodes defined as targets for the local node. For a multisystem node, the LIST keyword also displays information about the systems that make up the node. The LIST keyword is the default in the following situations:

- No other keyword is specified.
- Only the NODE(*) keyword is specified.
- Only the NODE(*nodename*) keyword is specified.
- Only the NODE(*nodename*) and SYSNAME(*sysname*) keywords are specified.
- Only the NODE(*nodename*) and SYSNAME(*) keywords are specified.
- Only the NODE(*) and SYSNAME(*) keywords are specified.

You can request detailed information for one target node or for all target nodes by specifying the NODE keyword. If you do not specify the NODE keyword, RACF lists a summary of all the target nodes defined for the local node.

The TARGET LIST command for a node displays the values last specified for the workspace data sets on a TARGET command for the node, but these values are not necessarily the values that are in effect. See “If You choose to let RACF allocate the VSAM data sets” on page 136 for information on why the values shown might not be the values in effect.

You can issue the TARGET LIST command to display either a summary or detailed list of information as shown in Table 7 on page 162.

Table 7. Displaying list information with the TARGET command

Command	Results
TARGET	Displays a summary list of all RRSF nodes known to the system on which the command is issued. For a multisystem node, each member system is listed.
TARGET LIST	
TARGET NODE(*)	Displays a detailed list of all RRSF nodes known to the system on which the command is issued. For a multisystem node, each member system is listed.
TARGET NODE(*) SYSNAME(*)	
TARGET NODE(*nodename*)	• If *nodename* is a single-system RRSF node, displays a detailed list of all RRSF nodes known to *nodename* • If *nodename* is a multisystem node, displays a summary list of its member systems
TARGET NODE(*nodename*) SYSNAME(*)	• If *nodename* is a single-system RRSF node, issues an error message • If *nodename* is a multisystem node, displays a detailed list of its member systems
TARGET NODE(*nodename*) SYSNAME(*sysname*)	Displays a detailed list of the named system in the named multisystem node

Sample output for single-system nodes: In this example, NODE1 is defined as the local node and NODE2, NODE3, NODE4, and RSFNODE4 are defined as its target nodes. Figure 17 illustrates the summary information displayed for a TARGET LIST command.

```
00- NODE1  IRRM009I (<) LOCAL RRSF NODE NODE1 IS IN THE OPERATIVE
  -  ACTIVE STATE.
  - IRRM009I (<) REMOTE RRSF NODE NODE2 IS IN THE OPERATIVE ACTIVE
  -  STATE
  - IRRM009I (<) REMOTE RRSF NODE NODE3 IS IN THE OPERATIVE PENDING
  -              CONNECTION STATE.
  - IRRM009I (<) REMOTE RRSF NODE NODE4 IS IN THE OPERATIVE PENDING
  -              CONNECTION STATE.
  - IRRM009I (<) REMOTE RRSF NODE RSFNODE4 IS IN THE OPERATIVE PENDING
  -              CONNECTION STATE.
```

Figure 17. Summary information displayed by a TARGET LIST command for a single-system node

Figure 18 on page 163 illustrates the detailed information displayed for a `TARGET NODE(NODE1) LIST` command.

```
IRRM010I (<) RSFJ SUBSYSTEM PROPERTIES OF LOCAL RRSF NODE NODE1:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - <NOT SPECIFIED>
   PROTOCOL    - APPC
                 LU NAME          - MF1AP001
                 TP PROFILE NAME  - IRRRACF
                 MODENAME         - <NOT SPECIFIED>
   TIME OF LAST TRANSMISSION TO   - <NONE>
   TIME OF LAST TRANSMISSION FROM - <NONE>
   WORKSPACE FILE SPECIFICATION
         PREFIX                   - "RSFJ.WORK"
         WDSQUAL                  - <NOT SPECIFIED>
         FILESIZE                 - 500
         VOLUME                   - TEMP01
         FILE USAGE
                 "RSFJ.WORK.NODE1.INMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
                 "RSFJ.WORK.NODE1.OUTMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
```

Figure 18. Detailed information displayed by a TARGET LIST command for a single-system node

Sample output for multisystem nodes: Figure 19 shows an example of the summary information that might be displayed for a multisystem node named NODE2. It contains two systems names MVSA and MVSB. The TARGET LIST command was issued from MVSA, which is the main system on NODE2. NODE1 is defined as the local node and NODE2, NODE3, NODE4, and RSFNODE4 are defined as its target nodes. The TARGET LIST command was issued from NODE NODE1.

```
IRRM009I (@) REMOTE RRSF NODE NODE2 SYSNAME MVSA IS IN THE OPERATIVE
  -  ACTIVE STATE.
  - IRRM009I (@) REMOTE RRSF NODE NODE2 SYSNAME MVSB IS IN THE OPERATIVE
  -  ACTIVE STATE
  - IRRM009I (@) REMOTE RRSF NODE NODE3 SYSNAME MVSX IS IN THE OPERATIVE
  -  PENDING CONNECTION STATE.
  - IRRM009I (@) REMOTE RRSF NODE NODE4 SYSNAME MVSY IS IN THE OPERATIVE
  -  PENDING CONNECTION STATE.
```

Figure 19. Summary information from the TARGET LIST command for a multisystem node. The multisystem node is named NODE2.

Figure 20 on page 164 shows an example of the detailed information that might be displayed for a system on a multisystem node. In this example, the command TARGET NODE(NODE2) SYSNAME(MVSA) was issued from system MVSX on node NODE3.

```
IRRM010I (@) RACF SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2
SYSNAME MVSA
  STATE       - OPERATIVE ACTIVE
  DESCRIPTION - <NOT SPECIFIED>
  PROTOCOL    - APPC
                LU NAME          - LU08
                TP PROFILE NAME  - IRRRACF
                MODENAME         - <NOT SPECIFIED>
  TIME OF LAST TRANSMISSION TO   - <NONE>
  TIME OF LAST TRANSMISSION FROM - <NONE>
  WORKSPACE FILE SPECIFICATION
        PREFIX                   - SYS1
        WDSQUAL                  - <NOT SPECIFIED>
        FILESIZE                 - 500
        VOLUME                   - <NOT SPECIFIED>
        FILE USAGE
                 "SYS1.LU05.LU08.INMSG"
                                 - CONTAINS 0 RECORD(S)
                                 - OCCUPIES 1 EXTENT(S)
                 "SYS1.LU05.LU08.OUTMSG"
                                 - CONTAINS 4 RECORD(S)
                                 - OCCUPIES 1 EXTENT(S)
```

Figure 20. Detailed information from the TARGET LIST command for a system on a multisystem node

Controlling outgoing requests from the local node

The OPERATIVE and DORMANT keywords on the TARGET command control whether outgoing requests from the local node are sent immediately to the target node or held. Profiles in the RRSFDATA class control what outgoing requests can be sent.

Use the **OPERATIVE** keyword to request that a connection to a node be made active, or operative. When a connection becomes operative, all requests for the node (such as directed commands) that are held on the OUTMSG queue are sent to the node via the transport mechanism specified by the PROTOCOL keyword. As long as the connection remains operative, new requests for the node are sent when they are placed on the OUTMSG queue.

For example, to request that NODEA's connection with NODEB be made operative, issue the following command on NODEA:

```
TARGET NODE(NODEB) OPERATIVE
```

You must make the local node's connection with itself operative before you can make the connection with the remote nodes operative. You must specify the PREFIX keyword for a node, and either preallocate the workspace data sets for the node or specify the WORKSPACE keyword, before you can make the connection with the node operative. You must also specify the PROTOCOL keyword, except for the local node when it is operating in local mode. For example, if you issue the following commands for a remote node:

```
TARGET NODE(KINGSTON) PREFIX(RRSF) DESCRIPTION('KINGSTON, NY')
TARGET NODE(KINGSTON) WORKSPACE(VOLUME(123456))
TARGET NODE(KINGSTON) OPERATIVE
```

RACF issues an error message on the TARGET OPERATIVE command because you did not provide protocol information for the node KINGSTON. The following commands should succeed:

```
TARGET NODE(KINGSTON) PREFIX(RRSF) DESCRIPTION('KINGSTON, NY')
TARGET NODE(KINGSTON) WORKSPACE(VOLUME(123456))
TARGET NODE(KINGSTON) PROTOCOL(APPC(LUNAME(MVS01)))
TARGET NODE(KINGSTON) OPERATIVE
```

Be aware that when you issue a TARGET OPERATIVE command, RACF begins trying to make the connection operative, but this can take time. Until the connection reaches the *operative active* state (see Table 5 on page 133), RACF treats the connection as if it were dormant, and continues to queue requests for the remote node. You can use the TARGET LIST command to determine what state the connection is in.

Use the **DORMANT** keyword to define a connection to a node as inactive, or dormant. When a connection is dormant, RACF queues requests for the node (such as directed commands) in the OUTMSG workspace data set for the node. The requests are held until the connection is made operative.

For example, to define NODEA's connection with NODEB as dormant, issue the following command on NODEA:

```
TARGET NODE(NODEB) DORMANT
```

The OPERATIVE and DORMANT keywords are not processed for remote nodes if the local node is in the initial state.

Controlling incoming requests from remote nodes

A node can control *what* requests it sends to other nodes, but a node has no control over what requests other nodes send to it; it can control only *whether* requests are sent. The OPERATIVE and DORMANT keywords on the TARGET command control whether incoming requests from the remote node are sent immediately or held.

Use the OPERATIVE keyword to request that a connection to a node be made active, or operative. When a connection with a remote node is made operative, all requests for the local node (such as directed commands) that are held on the OUTMSG queue of the remote node are sent to the local node via the transport mechanism specified on the PROTOCOL keyword. As long as the connection remains operative, new requests from the remote node are sent when they are placed on the OUTMSG queue.

Use the DORMANT keyword to request that a connection to a node be made inactive, or dormant. When a connection is dormant, RACF queues requests from the remote node (such as directed commands) in the OUTMSG workspace data set on the remote node. The requests are held until the connection is made operative.

Purging a workspace data set

Use the **PURGE** keyword to purge all entries from the specified INMSG or OUTMSG workspace data set for the node specified on the NODE keyword. You must make the connection with the node dormant before you can purge a workspace data set for it. When you purge a workspace data set, requests that were saved in it are lost, and database inconsistencies might result. The PURGE keyword can be used for error recovery when there is erroneous data in a workspace data set causing RACF to fail.

Deleting a node

Use the **DELETE** keyword to delete a node from the set of known target nodes. After you delete a node, the local node and the deleted node can no longer send requests to each other. Before you can delete a node, the connection with the node

must be dormant or defined. When you delete a node, any workspace data sets that are allocated for the node are deallocated. If the workspace data sets are empty, they are also deleted.

There is an order you need to observe when deleting nodes:

- Before deleting the local node, delete all other target nodes.
- Before deleting the local main system on a multisystem node, delete all of its remote nodes.
- If the local node is a multisystem node, before deleting the local system whose SYSNAME matches the current CVTSNAME delete all other target nodes.
- Before deleting the MAIN system of a remote multisystem node, delete all other member systems of that multisystem node.

For example, to delete NODEC from the set of targets known to NODEA, issue the following commands on NODEA:

```
TARGET NODE(NODEC) DORMANT
TARGET NODE(NODEC) DELETE
```

Reconfiguring a multisystem node

After you have configured a multisystem node, you might want to reconfigure it to:

- Add a new system to the multisystem node
- Delete a system from the multisystem node
- Make a different system the main system

Adding a system to a multisystem node: To add a system to a multisystem node, update the RACF parameter libraries on all nodes to add the TARGET commands needed to add the new member system. On each system, enter the same commands from the console. (Instead of entering the commands from the console, you can create a RACF parameter library member containing the new TARGET commands, and issue a SET INCLUDE command from the console on each system.)

If the system being added to the multisystem node is not new to the network (for example, if you are joining an existing single-system node to a multisystem node), you must also delete the existing TARGET information for the system you are adding. Update the RACF parameter libraries on all nodes to remove the original TARGET commands for the single-system node. Then enter TARGET DELETE commands from the console to undo the original TARGET commands.

Deleting a system from a multisystem node: To delete a functioning member system from a multisystem node:

1. Stop all TSO/E and batch processing on the system being deleted to allow the workspace data sets to empty and to ensure that no new requests are added.
2. Enter TARGET DORMANT commands to make all existing conversations from and to the system being deleted dormant.
3. Delete the TARGET commands that refer to the system from the RACF parameter libraries on:
   - The system being deleted
   - The other systems in the multisystem node
   - All other systems in the RRSF network

   Then, on each system enter TARGET DELETE commands from the console to undo the original TARGET commands.

You cannot delete a main system if there are any nonmain systems in the node. You must delete the nonmain systems first.

Configuring a new main system in a multisystem node: If you configure a multisystem node in your RRSF network, at a later time you might decide that you need to reconfigure the multisystem node to make a different system the main system. This type of reconfiguration is complex, and it is not a good solution when a system drops or a connection is broken. Instead, when a system drops, re-IPL it. When a connection breaks, let requests and returned output queue up in the workspace data sets while you fix the connection. However, if you need to shift the RRSF workload performed by the main system to another system, you need to reconfigure the multisystem node. RACF allows you to reconfigure the multisystem node dynamically, without stopping the RACF subsystem on any system other than the original local main system, or losing any RRSF requests or returned output.

When you reconfigure a multisystem node, you cannot safely just delete node definitions on live systems and redefine them. You risk losing requests from automatic direction or password synchronization while the nodes are not defined. Instead, to configure a different main system in a multisystem node, follow these steps:

1. Drop TSO/E and JES on the original local main system.

   Doing this prevents any further RACF activity on the system, and so prevents RACF databases from becoming unsynchronized if you use automatic command direction.
2. On the original local main system, issue the RACF STOP command to stop the RACF subsystem.

   Doing this causes the system to attempt to finish up all outstanding work and to close its workspace data sets. New requests and returned output from remote nodes will queue up in their OUTMSG workspace data sets.
3. Make connections dormant:
   - On the local system that is to be the new main, issue a TARGET DORMANT command for its local connection. Also issue TARGET DORMANT commands to make all connections with remote nodes dormant.
   - On each remote node, issue TARGET DORMANT commands for the original and new main systems. Do not perform step 7 on page 168 until the INMSG files for the original and new main systems on each remote node have drained.

   Suggestion

   In this step and others where you must issue the same command or set of commands on every system (or every nonmain system, in this case) on a multisystem node, you can make the task a little easier by creating a RACF parameter library member, IRROPT*xx*, containing the commands. Then you can issue a `SET INCLUDE(`*xx*`)` command on each system instead of manually entering the commands.

   After you issue the TARGET DORMANT commands, requests in the INMSG workspace data sets are processed, and their output is queued up in the OUTMSG workspace data sets. New requests and returned output also queue up in the OUTMSG data sets. Issue TARGET LIST commands to verify that the INMSG data sets on the local node have been drained before you go on to the next step.

4. If the workspace data sets for the original main system and the new main system are not on shared DASD with a shared catalog, copy the workspace data sets for the original main system to DASD accessible to the new main system, using the same workspace data set names.
5. On the new main system, issue a TARGET MAIN command to make it the main system. For example, if the multisystem node name is NODEABC, and MVSB is the new main system, issue:

   ```
   TARGET NODE(NODEABC) SYSNAME(MVSB) MAIN
   ```

   This command causes RACF to transfer requests and returned output from the original main system's OUTMSG workspace data sets to the new main system's OUTMSG workspace data sets.

   If you have not specified the prefixes for the workspace data sets and the LU names for the member systems consistently in the TARGET commands that defined the local multisystem node, this step will fail. See "Defining RRSF nodes to RACF" on page 156.
6. Issue the same TARGET MAIN command that you issued in step 5 on each nonmain system on the local multisystem node. Issue this command on the original main system only if it is to remain in the multisystem node.
7. Issue TARGET LIST commands to verify that the INMSG data sets on the remote nodes have been drained before you perform this step.

   On each remote system (that is, all remote systems of all remote nodes), issue the same TARGET MAIN command that you issued in step 5.
8. On the new main system, issue TARGET OPERATIVE commands to make the connection with itself and all connections with remote nodes operative.
9. On each remote system (that is, all remote systems of all remote nodes), issue TARGET OPERATIVE commands for the original main (if it is to remain in the multisystem node) and new main systems.
10. Update the TARGET commands in the RACF parameter libraries for all systems on all nodes in the RRSF network to reflect the new main system.

    If you fail to update the RACF parameter library for a system, the next time that system has its RACF subsystem restarted or is IPLed, the original TARGET commands will be issued, and requests and returned output will accumulate in the wrong OUTMSG workspace data set. However, RACF will issue appropriate error messages and prevent communications.

    You can update the parameter libraries at the same time you issue the TARGET MAIN commands in steps 5, 6, and 7. This approach helps to ensure that no parameter library updates are missed.
11. If the original main system is still part of the multisystem node, (and assuming that you have updated its RACF parameter library as discussed in step 10) restart the RACF subsystem, TSO/E and JES on the original main system.

The RACF parameter library

The RACF parameter library is a partitioned data set you can create whose members contain standard sequences of configuration commands that define the RRSF network from the local node's point of view. Using JCL you can specify a RACF parameter library member to be processed automatically as part of the initialization of RRSF. You can also dynamically execute the commands in one or more parameter library members by issuing a RACF SET INCLUDE operator command.

Security considerations for the RACF parameter library: You should protect the RACF parameter library via a DATASET profile. If you have not defined the user ID for the RACF subsystem as trusted or privileged, make sure it has READ access to the RACF parameter library.

Note: No OPERCMDS authority check is performed for commands issued from the RACF parameter library, and these commands run with the authority of the RACF subsystem address space user ID.

Configuring RRSF without using the RACF parameter library: You are not required to provide a RACF parameter library. RRSF configuration commands can be issued manually by an operator. However, the configuration commands must be issued each time the RACF subsystem address space is started. This process can be tedious and error-prone, and is not recommended.

Attributes of the RACF parameter library: The RACF parameter library must have the following attributes:

- Partitioned data set
- Cannot be a partitioned data set extended (PDSE)
- RECFM = FB
- LRECL = 80
- BLKSIZE = a multiple of 80
- Must be cataloged in the master catalog or an ICF user catalog, or the volume serial number must be specified on the DD statement

Parameter library member names: The members of the RACF parameter library have names in the form IRROPT*xx*. The first six characters, IRROPT, are required by RACF. They are followed by a one- or two-character alphanumeric suffix that you can use to define a unique member name. Some examples of valid member names are: IRROPT1, IRROPT02, IRROPTA1, and IRROPTAB.

Commands that can be issued from the RACF parameter library: The RACF SET and TARGET commands can be issued from the RACF parameter library. In addition, most other RACF commands can be issued from the parameter library. See *z/OS Security Server RACF Command Language Reference* for information on whether specific RACF commands can be issued from the RACF parameter library.

The following commands can also be issued from the RACF parameter library:

- ALLOCATE
- IRRDPI00
- FREE

Command direction is not allowed from the RACF parameter library. If you specify the AT or ONLYAT keyword on a command in the parameter library, RACF issues an error message and the command fails.

Automatic command direction does not occur for commands issued from the RACF parameter library.

Commands that span multiple lines: You can include commands that span multiple lines in the RACF parameter library. Use a minus sign (–) preceded by a blank at the end of a line to indicate that the command continues on the next line. A command in the RACF parameter library can be no longer than 20 lines of 72 characters each.

Comments in the RACF parameter library: You can include comments in the RACF parameter library only when they follow a command on a line. If you include a comment on a line without a command, RACF issues an error message. Comments are delimited by /* and */ characters.

Blank lines are not supported in the RACF parameter library.

Automatically processing a parameter library member during initialization: You can specify a member of the RACF parameter library to be automatically processed by the RACF initialization routine as part of the RACF subsystem address space initialization and program startup. The contents of this member are read by RACF and normally contain the RACF commands to configure the RRSF network from the local node's point of view.

Note: If you specify SMS workspace information for any node, there might be a delay during the processing of the parameter library member if the SMS address space is not initialized.

Specify the name of the RACF parameter library partitioned data set on the RACFPARM DD statement in the RACF procedure in SYS1.PROCLIB.

You can use SYS1.PARMLIB for the RACF parameter library. However, it is likely that different sets of users will need update authority to SYS1.PARMLIB and the RACF parameter library, and in these cases you should use different data sets.

Specify the RACF parameter library member to be processed by specifying its suffix on the `PARM='OPT=xx'` parameter on the EXEC statement in the RACF procedure in SYS1.PROCLIB. If you do not specify a suffix, it defaults to `00`. (See "Parameter library member names" on page 169 for information on the format of parameter library member names.)

After you update the RACF parameter library and the RACFPARM DD statement, if you want the configuration to take effect immediately do one of the following:

- Re-IPL MVS.
- Stop and restart the RACF subsystem address space, as shown:

```
subsystem_prefixSTOP
START subsystem-name,SUB=MSTR
```

Figure 21 on page 171 shows an example of a parameter library member that is automatically processed during initialization. In this case, the parameter library member IRROPT01 is processed to configure the local node, but no remote nodes are configured. Therefore, after initialization, NODEA is in local mode. Two other parameter library members, IRROPT02 and IRROPT03, exist but are not processed.

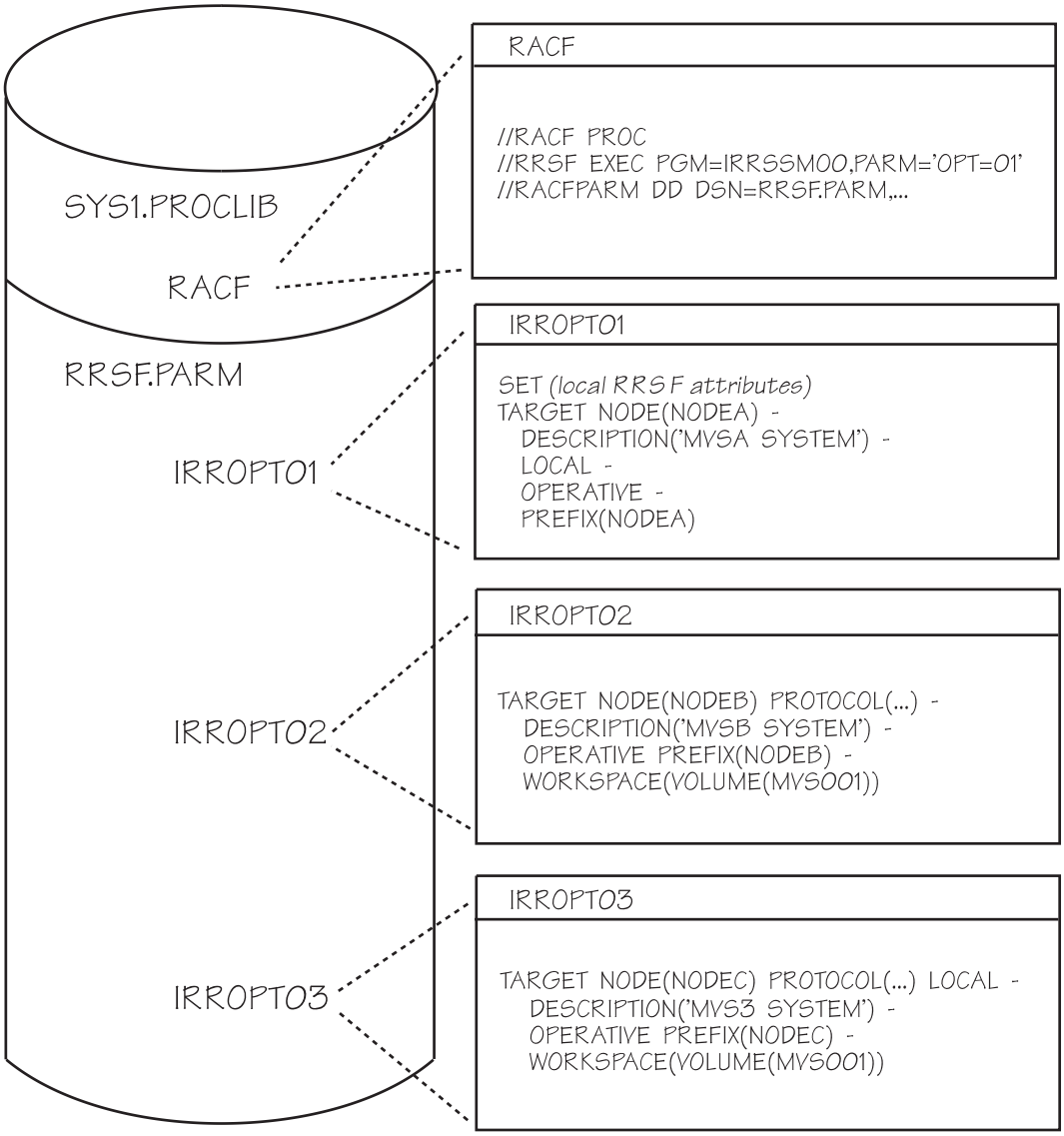


Figure 21. Example of a RACF parameter library for a node running in local mode

If you don't want a parameter library member processed automatically: You might want to use a RACF parameter library, but not have a member automatically processed during initialization. In that case, be aware that if you include a RACFPARM DD statement in your JCL, RACF assumes that you want a member processed automatically. If you do not specify a member on the PARM='OPT=xx' parameter, RACF attempts to process member IRROPT00 during initialization. If you do not have an IRROPT00 member, RACF issues a message. If you want to prevent RACF from automatically processing the IRROPT00 member in this situation, don't create one.

Using the SET INCLUDE function: The SET INCLUDE command allows you to specify a RACF parameter member to be processed, providing you with flexibility in your configuration process. You can issue the SET INCLUDE command as a RACF operator command, to dynamically change your configuration without having to manually enter the configuration commands. Or you can issue the SET INCLUDE command from a RACF parameter library member, to cause another member to be processed.

Note that you can issue a SET INCLUDE command to process a member that contains commands that you do not have authority to issue directly.

If you are using the RACF parameter library shown in Figure 21 on page 171, and you decide after initialization that you want NODEA to enter remote mode, you can enter the following RACF operator commands:

```
SET INCLUDE(02)
SET INCLUDE(03)
```

or

```
SET INCLUDE (02,03)
```

These commands cause RACF to process IRROPT02 and IRROPT03, configuring NODEA to communicate with NODEB and NODEC. If you decide that you want NODEA to always operate in remote node, you can add the SET INCLUDE commands to IRROPT01, as shown in Figure 22.

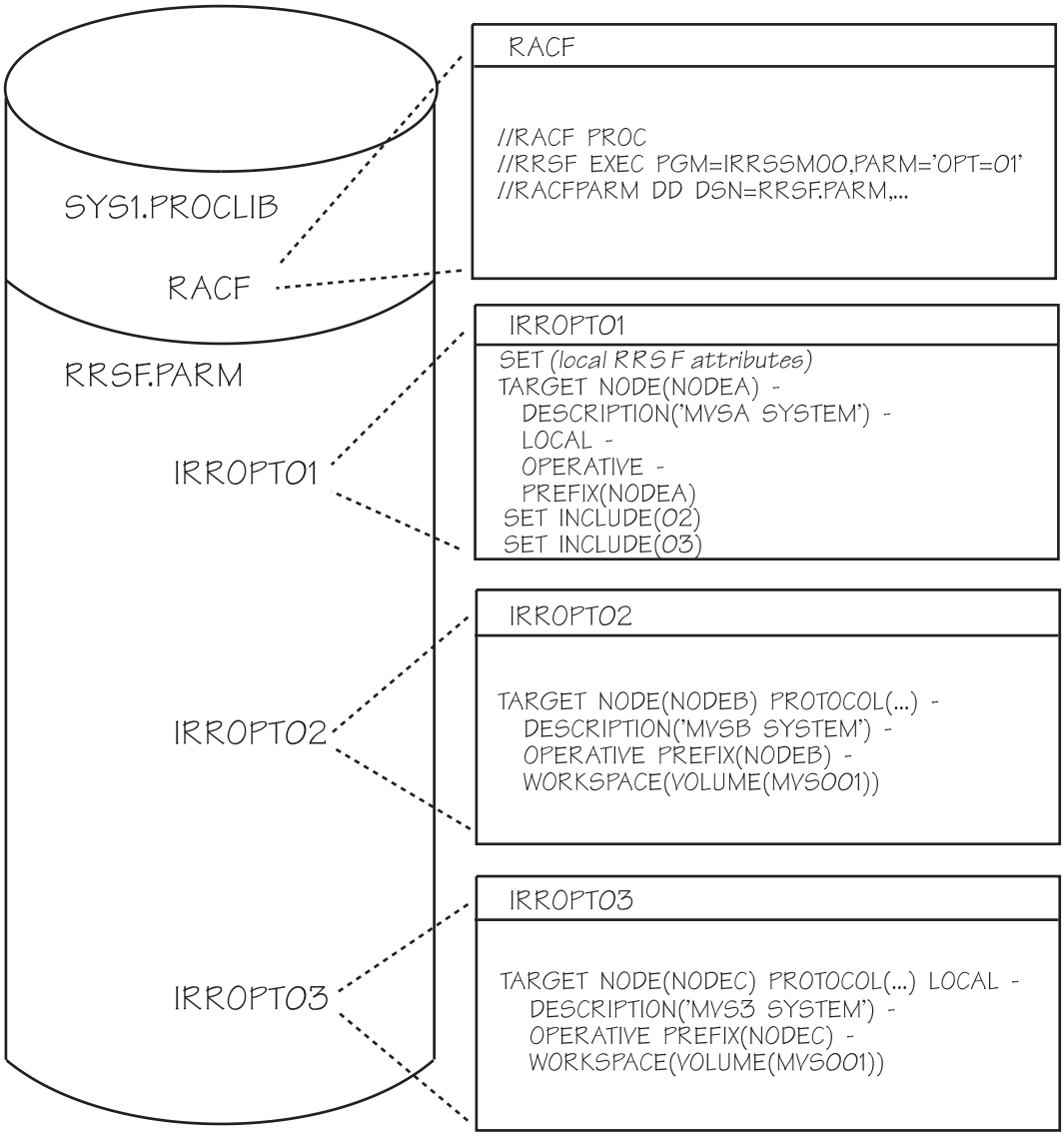


Figure 22. Example of a RACF parameter library for a node running in remote mode

A parameter library member that is included in another one via SET INCLUDE can in turn include another one, forming a hierarchy of included members. There is no limit on the number of levels of inclusion, but an included member cannot include a member that included it, or any other higher member in the inclusion hierarchy. Without this restriction, a never-ending loop of cyclic inclusion could occur. For example, in Figure 22 on page 172 you cannot code `SET INCLUDE(01)` in the IRROPT02 and IRROPT03 members. If you do, RACF issues an error message.

Sharing a RACF parameter library on a multisystem node: Although the member systems of a multisystem node do not communicate with each other via RRSF, each system must issue TARGET commands to define all of the systems in the multisystem node, and to identify the main system on the multisystem node. (RACF requires these commands to allow you, at a later time, to reconfigure the multisystem node with a different main system.) The commands can be issued using a single RACF parameter library that is shared by all of the systems on the multisystem node, and that contains all of the TARGET commands required by all of them. When RACF executes a TARGET command for the local node that includes the SYSNAME keyword, it compares the SYSNAME specified on the TARGET command with the CVTSNAME for the system the command is to run on. If the SYSNAME does not match the CVTSNAME, RACF does not process the OPERATIVE or DORMANT keyword. In addition, RACF issues an informational message and places it in the SYSLOG. This message might help diagnose why an expected conversation was not established.

For example, in the example shown in “Configuring a multisystem node” on page 179, a multisystem node named NODEAB has two member systems, MVSA and MVSB. MVSA is the main system. This node communicates via RRSF with a single-system RRSF node named NODEX. MVSA and MVSB could share a RACF parameter library containing the following member, to initialize their RRSF communications:

```
TARGET NODE(NODEAB) SYSNAME(MVSA) PROTOCOL(APPC(LUNAME(LU0A)))
       LOCAL WORKSPACE(VOLUME(MVS001)) PREFIX(SYS1) OPERATIVE MAIN

TARGET NODE(NODEAB) SYSNAME(MVSB) PROTOCOL(APPC(LUNAME(LU0B)))
       LOCAL WORKSPACE(VOLUME(MVS001)) PREFIX(SYS1) OPERATIVE

TARGET NODE(NODEX) PROTOCOL(APPC(LUNAME(LU0X)))
       WORKSPACE(VOLUME(MVS001)) PREFIX(SYS1) OPERATIVE
```

When this parameter library member runs on MVSA, the SYSNAME on the first TARGET LOCAL command matches the CVTSNAME of MVSA, and the local connection for MVSA is made operative. The SYSNAME on the second TARGET LOCAL command does not match the CVTSNAME of MVSA, so the OPERATIVE keyword is not processed, and the connection state from MVSA to MVSB is set to *defined*. RACF writes a message to the SYSLOG indicating that the OPERATIVE keyword was ignored. This message is expected, and you do not need to take any action. The TARGET command for NODEX runs and initiates an operative connection between MVSA and NODEX.

When this parameter library member runs on MVSB, the SYSNAME on the first TARGET LOCAL command does not match the CVTSNAME of MVSB, and the connection state from MVSB to MVSA is set to *defined*. RACF writes a message to the SYSLOG indicating that the OPERATIVE keyword was ignored. The SYSNAME on the second TARGET LOCAL command matches the CVTSNAME of MVSB, and the local connection for MVSB is made operative. The TARGET command for NODEX runs and an operative connection is initiated between MVSB and NODEX.

In order to share the RACF parameter library between systems, you must define it on shared DASD.

Order of commands in a RACF parameter library: If your RRSF network contains multisystem nodes, the order in which you issue TARGET DORMANT and TARGET OPERATIVE commands is important. For best results, when creating a RACF parameter library member issue all TARGET LOCAL commands before TARGET commands for remote connections, and issue all TARGET MAIN commands before TARGET commands for nonmain systems. Doing this helps to ensure that conversations are started.

Recovering from RACF parameter library errors: See "Recovering from RACF Parameter Library Problems" on page 327 for information on recovery for error conditions for the RACF parameter library.

Customizing and establishing security for RRSF

Use the RRSFDATA class to customize which functions will be available in your remote sharing environment, and to establish security for those functions.

Customizing a remote sharing environment

RACF provides you with flexibility in customizing the RACF remote sharing facility environment on each RRSF node. You can choose to allow some functions in your environment, and not allow others, or to restrict some functions to specific nodes. For example, you can choose to allow or not allow automatic command direction on an RRSF node, and if you choose to allow it you can choose which commands are automatically directed and to which nodes they are directed.

You can also control which user IDs are able to use each function. See "Establishing security for your remote sharing environment" on page 176 for information.

You customize the RACF remote sharing facility environment for an RRSF node by defining profiles in the RRSFDATA class. The customization can be done by either a system programmer or a security administrator.

The RRSFDATA class is a crucial class for RACF remote sharing. This class *must be active* on an RRSF node before you can use many of the functions of RRSF, including defining associations, synchronizing passwords, directing commands with the AT keyword, and automatic direction. The RRSFDATA class can be used as a switch to turn these remote sharing functions on and off as you activate and deactivate the class.

Note: We strongly recommend that you RACLIST the RRSFDATA class.

Table 8 on page 175 shows the RRSFDATA resource names and the remote sharing functions that they control.

Table 8. RRSFDATA resource names. The node name on a resource name is the name defined for a node by the TARGET command. For more information on defining node names, See "Configuring an RRSF network" on page 152.

Resource Name	Controls Authorization To...
AUTODASD.*node*.DATASET.APPL	Have RACF automatically direct RACROUTE REQUEST=DEFINE and RACDEF updates to DASD profiles in the DATASET class to node *node*. In most circumstances you should not set up automatic direction for these updates. For information on why you should not, see "Automatic direction of application updates" on page 125.
AUTODIRECT.*node.class*.APPL	Have RACF automatically direct application updates in class *class* to node *node*. In the DATASET class, only updates made by ICHEINTY, RACROUTE REQUEST=EXTRACT, and RACXTRT are covered by this resource name.
AUTODIRECT.*node.class.command*	Have RACF automatically direct all *command* commands in class *class* to node *node*.
AUTODIRECT.*node*.USER.PWSYNC	Have RACF automatically direct all password changes to node *node*.
AUTOTAPE.*node*.DATASET.APPL	Have RACF automatically direct RACROUTE REQUEST=DEFINE and RACDEF updates to tape profiles in the DATASET class to node *node*.
DIRECT.*node*	Specify the AT keyword on RACF commands to direct them to node *node*.
IRRBRW00	Execute the workspace data set VSAM file browser, IRRBRW00.
PWSYNC	Synchronize passwords with another user ID after establishing an association with that user ID that specifies password synchronization.
RACLINK.DEFINE.*node*	Issue the RACLINK DEFINE command to define an association with a user ID on node *node*.
RACLINK.PWSYNC.*node*	Issue the RACLINK DEFINE command to define an association that synchronizes passwords with a user ID on node *node*.

Initially, the RRSFDATA class is not active, and no profiles are defined in the class. Therefore the RRSF functions controlled by the RRSFDATA class are not available to any users. You must define profiles for the functions you want to use, and activate the RRSFDATA class to make the functions available. If you define a profile with UACC(READ), then all users by default have access to the function the profile controls. If you define a profile with UACC(NONE), then no users have access by default to the function the profile controls, and you must explicitly authorize users to use the function. (See "Establishing security for your remote sharing environment" on page 176.)

If you want, for example, to customize your network so that all user IDs on NODEA can define associations with user IDs on NODEB and direct commands to NODEB, but you don't want user IDs on NODEA to automatically synchronize their passwords with user IDs on NODEB, then on NODEA issue:

```
RDEFINE RRSFDATA RACLINK.DEFINE.NODEB UACC(READ)
RDEFINE RRSFDATA DIRECT.NODEB UACC(READ)
```

and then activate the RRSFDATA class:

```
SETROPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA)
```

Because there is no RRSFDATA profile for RACLINK.PWSYNC.NODEB, password changes made on NODEA will not be propagated to NODEB.

Security checks based on the RRSFDATA class are performed only on the local node, not on the remote nodes. So, for example, you can use the RRSFDATA class on NODEA to prevent users on NODEA from directing commands to NODEB, but the RRSFDATA class on NODEA cannot prevent users on NODEB from directing commands to NODEA. However, you can use the RRSFDATA class on NODEB to prevent users on NODEB from directing commands to NODEA.

For more information on RRSFDATA profiles, see *z/OS Security Server RACF Security Administrator's Guide*.

Establishing security for your remote sharing environment

After you customize the RACF remote sharing facility environment on an RRSF node by defining RRSFDATA profiles (see "Customizing a remote sharing environment" on page 174), the security administrator can control which users have access to which RRSF functions by granting or denying access to the RRSFDATA profiles. For example, if you have customized the RRSF environment on NODEA with the command

```
RDEFINE RRSFDATA DIRECT.NODEB UACC(NONE)
```

to not allow command direction to NODEB, then the security administrator can allow user ID WALT on NODEA to direct commands to NODEB by issuing the following command on NODEA:

```
PERMIT DIRECT.NODEB CLASS(RRSFDATA) ID(WALT) ACCESS(READ)
```

For more information on establishing security for an RACF remote sharing facility environment, see *z/OS Security Server RACF Security Administrator's Guide*.

Examples of defining a remote sharing environment

Following are some examples illustrating ways to define a remote sharing environment. For more examples, see member RACPARM in SYS1.SAMPLIB. See also Appendix B, "RRSF Initialization Worksheet and Scenario," on page 349.

Note: The commands shown in this section are for illustrative purposes only, and the syntax shown might change depending on how the commands are issued. For example, commands issued from the RACF parameter library require a - continuation character if they exceed a line, and commands issued as operator commands require a command prefix. See *z/OS Security Server RACF Command Language Reference* for details.

Configuring nodes in local mode

Assume that you have two nodes, NODEA and NODEB, and that you want to configure them each in local mode, as illustrated in Figure 23 on page 177.

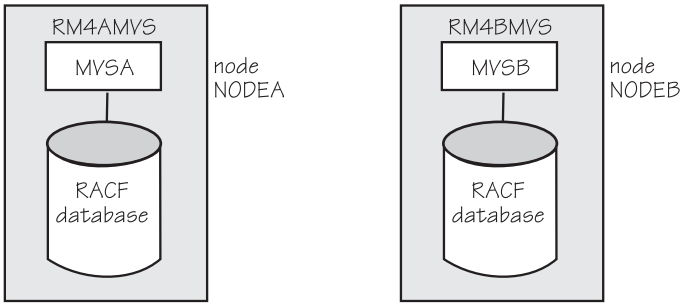


Figure 23. Two RRSF nodes in local mode

To configure NODEA as an RRSF node in local mode, issue the following command on NODEA:

```
TARGET NODE(NODEA)
       LOCAL
       DESCRIPTION('First sample node')
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

To configure NODEB as an RRSF node in local mode, issue the following command on NODEB:

```
TARGET NODE(NODEB)
       LOCAL
       DESCRIPTION('Second sample node')
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

Because the two nodes are in local mode, they do not require APPC, so you do not need the PROTOCOL keyword on the TARGET commands.

To customize NODEA so that:

- Users with multiple user IDs on NODEA can define user ID associations to synchronize their passwords.
- Users on NODEA who have the appropriate associations defined can synchronize their passwords.
- Users on NODEA cannot use the AT command to direct commands to other user IDs on NODEA.

enter the following commands on NODEA:

```
RDEFINE RRSFDATA RACLINK.DEFINE.NODEA UACC(READ)
RDEFINE RRSFDATA RACLINK.PWSYNC.NODEA UACC(READ)
RDEFINE RRSFDATA PWSYNC UACC(READ)
SETROPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA) GENERIC(RRSFDATA)
```

Because there is no RRSFDATA profile for DIRECT.NODEA, users on NODEA cannot direct commands to other user IDs on NODEA.

To customize NODEB so that:

- Users on NODEB cannot define user ID associations.
- Users on NODEB who have the appropriate associations defined (by the security administrator, for example) can synchronize their passwords with their other user IDs on NODEB.

- Users on NODEB who have the appropriate associations defined can direct commands to their other user IDs on NODEB.

enter the following commands on NODEB:

```
SETROPTS GENERIC(RRSFDATA)
RDEFINE RRSFDATA RACLINK.DEFINE.* UACC(NONE)
RDEFINE RRSFDATA PWSYNC UACC(READ)
RDEFINE RRSFDATA DIRECT.* UACC(READ)
SETROPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA)
```

Configuring a two-node network

Figure 24 shows an RRSF network with two nodes, NODEA and NODEB. Because NODEA and NODEB are to communicate with each other, they must be configured in remote mode. The ACBNAME in the APPCPM*xx* member of SYS1.PARMLIB on NODEA is RM4AMVS. The ACBNAME in the APPCPM*xx* member of SYS1.PARMLIB on NODEB is RM4BMVS.

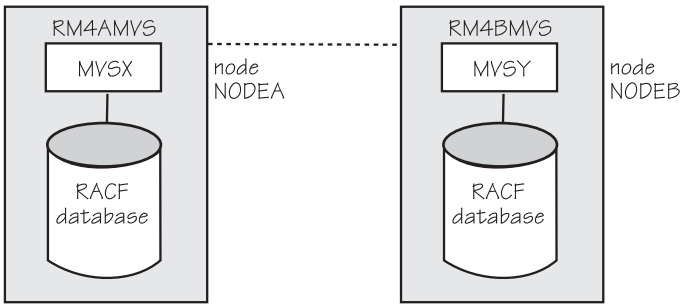


Figure 24. An RRSF network with two nodes

The following commands illustrate how you could configure this network. On NODEA the following command describes the local node NODEA:

```
TARGET NODE(NODEA)
       LOCAL
       DESCRIPTION('First sample node')
       PROTOCOL(APPC(LUNAME(RM4AMVS)))
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

and the following command describes NODEB as a remote node of NODEA:

```
TARGET NODE(NODEB)
       DESCRIPTION('Second sample node')
       PROTOCOL(APPC(LUNAME(RM4BMVS)))
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

On NODEB the following command describes the local node NODEB:

```
TARGET NODE(NODEB)
       LOCAL
       DESCRIPTION('Second sample node')
       PROTOCOL(APPC(LUNAME(RM4BMVS)))
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

and the following command describes NODEA as a remote node of NODEB:

```
TARGET NODE(NODEA)
       DESCRIPTION('First sample node')
       PROTOCOL(APPC(LUNAME(RM4AMVS)))
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

To customize both nodes to allow users to create user ID associations and synchronize passwords with user IDs on either node, but not to direct commands, enter the following commands on each node:

```
SETROPTS GENERIC(RRSFDATA)
RDEFINE RRSFDATA RACLINK.DEFINE.* UACC(READ)
RDEFINE RRSFDATA RACLINK.PWSYNC.* UACC(READ)
RDEFINE RRSFDATA PWSYNC UACC(READ)
RDEFINE RRSFDATA DIRECT.* UACC(NONE)
SETROPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA)
```

Configuring a multisystem node

The following scenario illustrates how you could configure the RRSF network shown in Figure 25, and customize the nodes to use automatic direction to synchronize changes made to their USER and GROUP profiles.

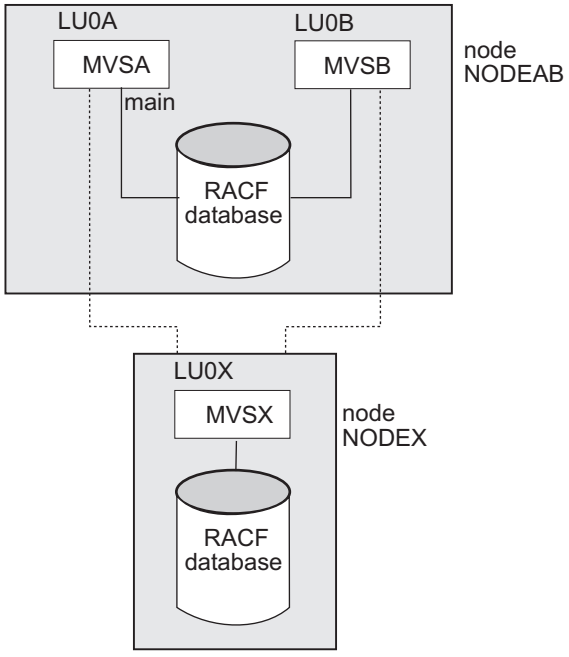


Figure 25. An RRSF network containing a multisystem node and a single-system node. NODEAB is a multisystem node. NODEX is a single system node.

In this network, MVSA and MVSB share a RACF database, and are configured as member systems of multisystem node NODEAB. MVSA is the main system on NODEAB. MVSX has its own RACF database, which it does not share with any other system, and is configured as a single-system RRSF node, NODEX. APPC/MVS is active on all nodes. MVSA has the LU name LU0A, MVSB has the LU name LU0B, and MVSX has the LU name LU0X. The dotted lines represent RRSF communication.

Steps for configuring a multisystem node:

Before you begin: You need to have the information about the nodes. You should use the information from Appendix B, "RRSF Initialization Worksheet and Scenario," on page 349.

Perform the following steps to configure a multisystem node.

1. Synchronize the USER and GROUP profiles between the two databases. For information on how you can do this, see "Synchronizing database profiles" on page 145.

---

2. On MVSA and MVSB, issue RACF commands to configure MVSA and MVSB as member systems of multisystem node NODEAB.

   Example: On MVSA and MVSB, issue:

   ```
   TARGET NODE(NODEAB) SYSNAME(MVSA) PROTOCOL(APPC(LUNAME(LU0A)))
          LOCAL WORKSPACE(VOLUME(MVS001)) PREFIX(SYS1) OPERATIVE MAIN

   TARGET NODE(NODEAB) SYSNAME(MVSB) PROTOCOL(APPC(LUNAME(LU0B)))
          LOCAL WORKSPACE(VOLUME(MVS001)) PREFIX(SYS1) OPERATIVE
   ```

   Create a shared RACF parameter library for MVSA and MVSB, and set up MVSA and MVSB to invoke a common member of the parameter library at initialization. (See "The RACF parameter library" on page 168 for information on how to do this.) Add the commands you just issued to the common initialization member, to ensure that in the future RACF comes up with the correct RRSF configuration.

---

3. On MVSA and MVSB, issue RACF commands to configure NODEX as a remote RRSF node for each of them.

   Example: On MVSA and MVSB, issue:

   ```
   TARGET NODE(NODEX) PROTOCOL(APPC(LUNAME(LU0X)))
          WORKSPACE(VOLUME(MVS001)) PREFIX(SYS1) OPERATIVE
   ```

   Add this command to the common initialization member of the RACF parameter library.

---

4. On NODEX, issue RACF commands to define NODEAB as a remote multisystem node for NODEX, with MVSA defined as the main system.

   Example: On NODEX, issue:

   ```
   TARGET NODE(NODEX) PROTOCOL(APPC(LUNAME(LU0X))) LOCAL
          WORKSPACE(VOLUME(MVS005)) PREFIX(SYS1) OPERATIVE

   TARGET NODE(NODEAB) SYSNAME(MVSA) PROTOCOL(APPC(LUNAME(LU0A))) MAIN
          WORKSPACE(VOLUME(MVS005)) PREFIX(SYS1) OPERATIVE

   TARGET NODE(NODEAB) SYSNAME(MVSB) PROTOCOL(APPC(LUNAME(LU0B)))
          WORKSPACE(VOLUME(MVS005)) PREFIX(SYS1) OPERATIVE
   ```

   Create a RACF parameter library for NODEX and set up NODEX to invoke a member of the parameter library at initialization. (See "The RACF parameter library" on page 168 for information on how to do this.) Add the commands you just issued to the initialization member, to ensure that in the future RACF comes up with the correct RRSF configuration.

---

5. Define profiles in the RRSFDATA class to use automatic direction to synchronize changes made to USER and GROUP profiles on NODEAB and NODEX. Issue SET commands on both nodes to activate automatic direction and specify the user ID and node to which notification of and output from failures in automatic direction are to be directed. Issue SETROPTS commands on both nodes to refresh the RRSFDATA profiles.

   Examples: On either MVSA or MVSB, issue:

   ```
   RDEFINE RRSFDATA AUTODIRECT.NODEX.USER.* UACC(READ)
   RDEFINE RRSFDATA AUTODIRECT.NODEX.GROUP.* UACC(READ)
   RDEFINE RRSFDATA AUTODIRECT.NODEX.USER.PWSYNC UACC(READ)
   ```

   On MVSA issue:

   ```
   SET AUTODIRECT (OUTPUT(FAIL(NODEX.ADMIN)) NOTIFY(FAIL(NODEX.ADMIN)))
   SET AUTOAPPL (OUTPUT(FAIL(NODEX.ADMIN)) NOTIFY(FAIL(NODEX.ADMIN)))
   SETR RACLIST(RRSFDATA) REFR
   ```

   On MVSB issue:

   ```
   SET AUTODIRECT (OUTPUT(FAIL(NODEX.ADMIN)) NOTIFY(FAIL(NODEX.ADMIN)))
   SET AUTOAPPL (OUTPUT(FAIL(NODEX.ADMIN)) NOTIFY(FAIL(NODEX.ADMIN)))
   SETR RACLIST(RRSFDATA) REFR
   ```

   On NODEX, issue:

   ```
   RDEFINE RRSFDATA AUTODIRECT.NODEAB.USER.* UACC(READ)
   RDEFINE RRSFDATA AUTODIRECT.NODEAB.GROUP.* UACC(READ)
   RDEFINE RRSFDATA AUTODIRECT.NODEAB.USER.PWSYNC UACC(READ)
   SET AUTODIRECT (OUTPUT(FAIL(NODEX.ADMIN)) NOTIFY(FAIL(NODEX.ADMIN)))
   SET AUTOAPPL (OUTPUT(FAIL(NODEX.ADMIN)) NOTIFY(FAIL(NODEX.ADMIN)))
   SETR RACLIST(RRSFDATA) REFR
   ```

You know you are done when automatic direction is activated.

Configuring two multisystem nodes

The following scenario illustrates how you could configure the RRSF network shown in Figure 26 on page 182, and customize the nodes to use automatic direction to synchronize changes made to their USER and GROUP profiles by TSO/E commands and application updates.

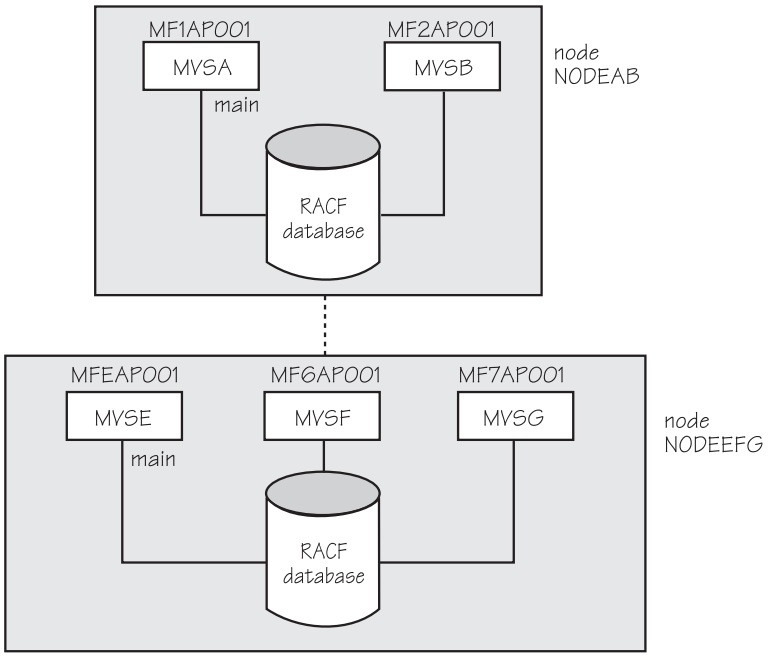


Figure 26. An RRSF network containing two multisystem nodes. NODEAB and NODEEFG are both multisystem nodes.

In this network, systems MVSA and MVSB share a RACF database, and are configured as member systems of multisystem node NODEAB. MVSA is the main system on NODEAB. Systems MVSE, MVSF, and MVSG share a RACF database, and are configured as member systems of multisystem node NODEEFG. MVSE is the main system on NODEEFG.

1. Create a shared RACF parameter library for MVSA and MVSB, and set up MVSA and MVSB to invoke a common member of the parameter library, IRROPTAB, at initialization. (See “The RACF parameter library” on page 168 for information on how to do this.) Add the following commands to IRROPTAB to configure MVSA and MVSB as member systems of multisystem node NODEAB:

```
TARGET NODE(NODEAB) SYSNAME(MVSA) LOCAL MAIN PREFIX(SYS1)
   PROTOCOL(APPC(LUNAME(MF1AP001)) WORKSPACE(VOLUME(TEMP01)) OPERATIVE
TARGET NODE(NODEAB) SYSNAME(MVSB) LOCAL PREFIX(SYS1)
   PROTOCOL(APPC(LUNAME(MF2AP001)) WORKSPACE(VOLUME(TEMP01)) OPERATIVE
```

2. Add the following commands to IRROPTAB to configure NODEEFG as a remote multisystem node for NODEAB:

```
TARGET NODE(NODEEFG) SYSNAME(MVSE) MAIN PREFIX(SYS3)
   PROTOCOL(APPC(LUNAME(MFEAP001)) WORKSPACE(VOLUME(TEMP01))
TARGET NODE(NODEEFG) SYSNAME(MVSF) PREFIX(SYS3)
   PROTOCOL(APPC(LUNAME(MF6AP001)) WORKSPACE(VOLUME(TEMP01))
TARGET NODE(NODEEFG) SYSNAME(MVSG) PREFIX(SYS3)
   PROTOCOL(APPC(LUNAME(MF7AP001)) WORKSPACE(VOLUME(TEMP01))
```

3. Create a shared RACF parameter library for MVSE, MVSF, and MVSG, and set up MVSE, MVSF, and MVSG to invoke a common member of the parameter library, IRROPTEG, at initialization. (See “The RACF parameter library” on page 168 for information on how to do this.) Add the following commands to IRROPTEG, to configure MVSE, MVSF, and MVSG as member systems of multisystem node NODEEFG:

```
TARGET NODE(NODEEFG) SYSNAME(MVSE) LOCAL MAIN PREFIX(MSN.SYS3)
   PROTOCOL(APPC(LUNAME(MFEAP001)) WORKSPACE(VOLUME(D79PK5))
TARGET NODE(NODEEFG) SYSNAME(MVSF) LOCAL PREFIX(MSN.SYS3)
```

```
   PROTOCOL(APPC(LUNAME(MF6AP001)) WORKSPACE(VOLUME(D79PK5))
TARGET NODE(NODEEFG) SYSNAME(MVSG) LOCAL PREFIX(MSN.SYS3)
   PROTOCOL(APPC(LUNAME(MF7AP001)) WORKSPACE(VOLUME(D79PK5))
```

4. Add the following commands to IRROPTEG to configure NODEAB as a remote multisystem node for NODEEFG:

```
TARGET NODE(NODEAB) SYSNAME(MVSA) MAIN PREFIX(MSN.SYS1)
   PROTOCOL(APPC(LUNAME(MF1AP001)) WORKSPACE(VOLUME(D79PK5)) OPERATIVE
TARGET NODE(NODEAB) SYSNAME(MVSB) PREFIX(MSN.SYS1)
   PROTOCOL(APPC(LUNAME(MF2AP001)) WORKSPACE(VOLUME(D79PK5)) OPERATIVE
```

5. On NODEAB, define profiles in the RRSFDATA class to use automatic direction to synchronize changes made via TSO/E commands and application updates to USER and GROUP profiles on MVSA and MVSB. On either MVSA or MVSB issue:

```
RDEFINE RRSFDATA AUTODIRECT.NODEEFG.USER.* UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEEFG.GROUP.* UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEEFG.USER.APPL UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEEFG.GROUP.APPL UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEEFG.USER.PWSYNC UACC(READ)
```

   Add the following commands to IRROPTAB to activate RRSF with automatic direction active.

```
SET AUTODIRECT (OUTPUT(FAIL(NODEAB.ADMIN)) NOTIFY(FAIL(NODEAB.ADMIN)))
SET AUTOAPPL (OUTPUT(FAIL(NODEAB.ADMIN)) NOTIFY(FAIL(NODEAB.ADMIN)))
SET PWSYNC (OUTPUT(FAIL(NODEAB.ADMIN)) NOTIFY(FAIL(NODEAB.ADMIN)))
SETROPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA)
```

6. Define profiles in the RRSFDATA class to use automatic direction to synchronize changes made via TSO/E commands and application updates to USER and GROUP profiles on MVSE, MVSF, and MVSG. On one of MVSE, MVSF, and MVSG issue:

```
RDEFINE RRSFDATA AUTODIRECT.NODEAB.USER.* UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEAB.GROUP.* UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEAB.USER.APPL UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEAB.GROUP.APPL UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.NODEEFG.USER.PWSYNC UACC(READ)
```

   Add the following commands to IRROPTAB to activate RRSF with automatic direction active.

```
SET AUTODIRECT (OUTPUT(FAIL(NODEEFG.ADMIN)) NOTIFY(FAIL(NODEEFG.ADMIN)))
SET AUTOAPPL (OUTPUT(FAIL(NODEEFG.ADMIN)) NOTIFY(FAIL(NODEEFG.ADMIN)))
SET PWSYNC (OUTPUT(FAIL(NODEEFG.ADMIN)) NOTIFY(FAIL(NODEEFG.ADMIN)))
SETROPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA)
```

7. Issue the following command on systems MVSA and MVSB, to cause the commands in IRROPTAB to run:

```
SET INCLUDE(AB)
```

8. Issue the following command on systems MVSE, MVSF, and MVSG, to cause the commands in IRROPTEG to run:

```
SET INCLUDE(EG)
```

Monitoring your remote sharing environment

An RRSF network is a complex system. It is composed of many elements:

- The RACF Subsystem with all its restartable functions
- RRSF definitions and sequences
- VSAM workspace files
- RRSFLIST command output files

- The VTAM definitions and APPC/MVS connections between nodes

You need to plan carefully to correctly implement an RRSF network. Two ITSC Redbooks can help with this planning:

- *RACF Version 2 Release 2 Installation and Implementation Guide*
- *RACF Version 2 Release 2 Technical Presentation Guide*

Monitoring the RRSF environment is a recommended practice for maintaining a healthy network. An RRSF environment is comprised of many components and can have many physical nodes. At any time there might be nodes that are not operational because of scheduled maintenance or a known problem that is being addressed. Only the professionals charged with maintaining the RRSF network can determine if messages or command results are as expected or if they indicate a problem that must be investigated and resolved.

Some monitoring approaches for customers to consider are:

1. Periodically issue the TARGET LIST command to determine that the nodes you expect to be operative are in fact operative. Additionally, look for unexpected messages sent to the operator's console that indicate whether a connection's state has changed to an error state. For example, IRRC022I or IRRC033I indicate that the state changed to Operative Error or IRRC032I indicates Dormant Error.
2. Periodically issue the TARGET LIST NODE(*node name*) command for each node to check the status of the workspace data sets, For example, look for the number of records in the data sets. If the number is excessive, the data sets can fill up. If they fill up, requests might be rejected and database inconsistencies might occur. Further, look for messages indicating a problem with the workspace data sets. For example, IRRC029I and IRRC030I indicate problems in trying to write to workspace data sets and IRRC031I indicates that a workspace data set is full. If a workspace data set fills up, refer to Chapter 9, "Recovery Procedures," on page 303 for more information.
3. If you use automatic direction, enter the SET command with the OUTPUT option to put the output (at least FAIL output) for automatically directed commands, automatically directed passwords, synchronized passwords, or automatically directed application updates into the RRSFLIST data set of the user responsible for maintaining RRSF. You should check the RRSFLIST data set periodically for unexpected results. Also, users must maintain their own RRSFLIST data set. To prevent it from filling up, move any results you need to another file and delete the contents. If the RRSFLIST data set fills up, output is sent via TSO TRANSMIT to that user.

   Guideline: Use the SET command with the NOTIFY option to specify at least one backup user to receive notification of whether the RRSF request is successful, in the event that the primary administrator is unavailable. If the users who should receive the RRSF command output or who receive notification are not logged on, significant storage could be consumed over time, since the output or results are queued for delivery or receipt when the user logs on. This storage consumption could result in additional system problems.
4. If explicit command direction (AT, ONLYAT) is commonly used, check the RRSFLIST data set of the command issuer for unexpected results.

Steps such as these allow for timely identification of problems you can correct before they become critical. See *z/OS Security Server RACF Diagnosis Guide* for detailed information on the setup necessary for RRSF and the errors possible with workspace data sets, APPC communications, and RRSF definitions and sequences.

See “Failures in the RACF Subsystem Address Space” on page 327 for error recovery information. Understanding the kinds of problems that can occur is a first step in deciding on the procedures necessary to detect and handle them.

Chapter 6. The RACF/DB2 external security module

Installing the RACF/DB2 external security module 188
Customizing the RACF/DB2 external security module (optional) 189
Customizing the number of exit work area cells 190
Choosing the class scope 190
Single subsystem class scope (classification model I) 191
Multi-subsystem class scope (classification model II) 193
Defining classes for the RACF/DB2 external security module (optional) 194
DB2 object types. 195
Assembling and link-editing the RACF/DB2 external security module . . . 195
RACF/DB2 external security module functions 196
The initialization function (XAPLFUNC = 1) 196
Initialization return and reason codes 197
The authorization function (XAPLFUNC = 2) 197
Authorization checking return and reason codes 197
The termination function (XAPLFUNC = 3) 198
Termination return and reason codes 198

Attention

This section applies to using RACF with DB2 Version 7 and below. For information about using RACF with DB2 Version 8, see *DB2 RACF Access Control Module Guide*.

The RACF/DB2 external security module allows you to use RACF as an alternative to DB2 authorization checking for DB2 objects and authorities.

RACF support for the RACF/DB2 external security module includes:

- The RACF/DB2 external security module as a sample assembler language routine that is invoked at the DB2 access control authorization exit point. It is a replacement for the default routine shipped with the DB2 product.
- A set of 29 general resource classes in the class descriptor table (CDT) supplied by IBM, ICHRRCDX. They are used by the RACF/DB2 external security module if all options keep their default values.
- A set of 29 RACF router table entries in the RACF router table supplied by IBM, ICHRFR0X.
- Support for RACROUTE REQUEST=FASTAUTH,LOGSTR=. This support provides additional information on audit records created by the RACF/DB2 external security module.
- Support for RACROUTE REQUEST=FASTAUTH,ACEEALET=. This support allows the RACF/DB2 external security module to access ACEEs in address spaces other than the HOME address space.

The exit point associated with the RACF/DB2 external security module is a DB2 exit point, not a RACF exit point. DB2 provides a dummy exit as the default. If you want to use the external security module, you need to install the RACF/DB2 external security module and replace the dummy exit. For specific information about the exit point and its interface, see the DB2 publications.

© Copyright IBM Corp. 1994, 2004

Installing the RACF/DB2 external security module

The Security Server provides the RACF/DB2 external security module as an assembler source module. It resides in the IRR@XACS member of SYS1.SAMPLIB. Before you can use RACF with DB2 objects and authorities, you need to install the RACF/DB2 external security module using the following procedures:

1. Copy IRR@XACS to a private library, using DSNX@XAC as the member name. The '@' has a value of X'7C'.
2. Customize the RACF/DB2 external security module. This step is optional and is necessary only if you want to modify the customization options from their default values. For information on the customization options, see "Customizing the RACF/DB2 external security module (optional)" on page 189.
3. Define classes for the RACF/DB2 external security module. This step is optional and is necessary only when you have modified the customization options from their default values in the previous step. For information on defining classes, see "Defining classes for the RACF/DB2 external security module (optional)" on page 194.
4. When the previous steps have completed, the security administrator should define the appropriate profiles and activate the necessary classes. For more information, see *z/OS Security Server RACF Security Administrator's Guide*.

   IBM provides a sample utility, RACFDB2, that converts the contents of SYSIBM.SYS*xxx*AUTH tables to equivalent RACF profiles. IBM does not support the RACFDB2 utility. For information on how to get this tool and others from the RACF home page or via anonymous FTP, see "Internet sources" on page xviii.
5. Assemble and link-edit the RACF/DB2 external security module. For information about performing this step, see "Assembling and link-editing the RACF/DB2 external security module" on page 195.

> **Attention**
>
> Do this step *only* after you have defined the profiles, activated the classes, and completed all the setup. If no classes are active when the RACF/DB2 external security module is invoked at DB2 subsystem startup, RACF issues message IRR901A. This indicates to DB2 that no classes are active.

After you complete these steps, the RACF/DB2 external security module becomes active the next time the DB2 subsystem is started. DB2 can invoke the RACF/DB2 external security module as soon as it is active.

When DB2 invokes the RACF/DB2 external security module, you can use the information found in messages IRR908I through IRR911I to help you understand how it is set up for a particular subsystem. These messages identify the:

- Version and length of the RACF/DB2 external security module
- Name of the DB2 subsystem or group attach name
- RACF FMID or APAR number associated with the module
- Length of all CSECTs contained in the module
- Options used for the module
- Classes that the module is trying to use
- Classes for which a RACROUTE request was successful

Multiple subsystems

Multiple subsystems using the same customization options can share the same copy of the RACF/DB2 external security module. Additional copies are necessary only when the subsystems choose different options.

Customizing the RACF/DB2 external security module (optional)

This section defines the customization options and corresponding class name formats related to the RACF/DB2 external security module. The default values for the customization options are defined in Figure 27 on page 190. These values result in multi-subsystem scope classes with 'DSN' as the class root. For example, the MDSNTB class protects DB2 tables.

Using the default values allows the RACF/DB2 external security module to use the classes in the class descriptor table (CDT) supplied by IBM. When you modify the customization options from their default values, you need to define classes in the installation-supplied class descriptor table.

You can modify the way the external security module works using several assembler SET symbols. They are located in the top of the source data set for IRR@XACS.

Attention

Customizing the RACF/DB2 external security module is optional. It is required *only* when you do not use the defaults.

SET Symbol **Description**

&CLASSOPT Specifies the class scope option

Allowable values are:

1 single subsystem class scope

2 multi-subsystem class scope. This is the default.

&CLASSNMT Specifies the class name root, which is characters 2 - 5 of the class name. It must be from 1 to 4 characters long and is used only for &CLASSOPT='2'. The default value is 'DSN'.

&CHAROPT Specifies the class name suffix, which is the last character of the class name for installation-defined classes. Valid values are 0-9, #, @, or $. The default is '1'.

&ERROROPT Specifies, for DB2 Version 7 or later, the action to take in the event of a DB2 initialization or authorization error. The allowable values are:

1 Native DB2 authorization is used. This is the default.

2 The DB2 subsystem is requested to stop.

&PCELLCT Specifies the number of primary work area cells

&SCELLCT Specifies the number of secondary work area cells

&SERVICELEVEL

For IBM use only

Notes:

1. &CLASSOPT, &CLASSNMT, and &CHAROPT specify the format of the class names and resource names the RACF/DB2 external security module uses. They are described in more detail in "Choosing the class scope."
2. &PCELLCT and &SCELLCT specify the number of primary and secondary work area cells. They are described in "Customizing the number of exit work area cells."
3. &ERROROPT is effective only for DB2 Version 7 or later. The DB2 system can be stopped for one of the following reasons:
   - The exit no longer needs DB2 to call the exit.
   - The exit abends.
   - DB2 receives an unexpected return code (EXPLRC1).

The following example shows the default values shipped with the RACF/DB2 external security module:

```
    GBLC  &CLASSNMT,&CHAROPT,&CLASSOPT
                GBLA  &PCELLCT,&SCELLCT
&CLASSOPT     SETC  '2'     1 - Use Single Subsystem Class Scope
*                                 Classification Model I
*                                 (One set of classes for EACH subsys)
*                               2 - Use Multi-Subsystem Class Scope
*                                 Classification Model II
*                                 (One set of classes for ALL subsys)
&CLASSNMT     SETC  'DSN'
```

DB2 Subsystem Name (Up to 4 chars) &CHAROPT SETC '1' One character suffix (0-9, #, @ or $) &ERROROPT SETC '2' 1 - Use Native DB2 authorization * 2 - Stop the DB2 subsystem &PCELLCT SETA 50 Primary Cell Count &SCELLCT SETA 50 Secondary Cell Count

Figure 27. Default values for settable options

Customizing the number of exit work area cells

When you invoke the external security module, it uses CPOOL cells as a work area to contain local variables. When you invoke the external security module for initialization, it allocates a primary pool of work area cells to be used on authorization requests. Each time the external security module is invoked for an authorization request, it obtains a cell and returns it when processing completes. If there are no more cells available, it uses a secondary pool of cells. You can control the number of cells allocated in the primary and secondary cell pools with the &PCELLCT and &SCELLCT SET symbols.

Choosing the class scope

One general resource class is associated with each DB2 object type. You can define up to two classes for each object type and set them up as associated members or grouping classes. The list of supported DB2 objects and class abbreviations is defined in "DB2 object types" on page 195. If only one class is used for an object, it must be defined with the member prefix. An additional class is used to support DB2 administrative authorities. The format of the class names of DB2 objects depends on the classification model you use.

You can alter the &CLASSOPT field of the assembler source statement in the external security module to select the class scope you want to use:

- Single subsystem class scope (Classification model I)
- Multi-subsystem class scope (Classification model II)

After you select a class scope, you can use the &CLASSNMT and &CHAROPT SET symbols to alter the default naming conventions for the general resource classes. The naming conventions for the resource names are defined in *z/OS Security Server RACF Security Administrator's Guide*.

The RACF/DB2 external security module uses the values &CLASSOPT, &CLASSNMT, and &CHAROPT to determine the format of the class names and resource names used to protect the DB2 objects.

Attention

The &CLASSOPT, &CLASSNMT, &CHAROPT, and &ERROROPT options are global in scope for each DB2 subsystem and must be the same for all classes. You can set the RACF/DB2 external security module to process only one class scope for each occurrence.

Single subsystem class scope (classification model I)

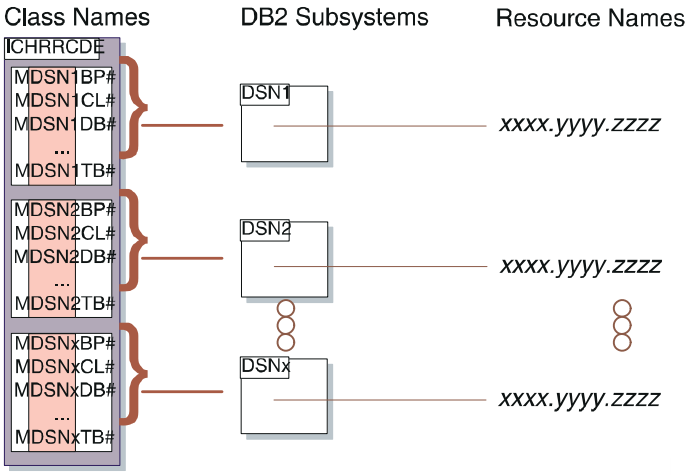


Figure 28. Single subsystem scope classes

When you choose this model, the RACF/DB2 external security module places the DB2 subsystem name in the class name. The format of a general resource class name associated with this DB2 object type is:

ayyyyxxz

Where

a is M for member class or G for grouping class

yyyy is the DB2 subsystem or DB2 group attach name (from XAPLGPAT)

xx is the type of DB2 object. See “DB2 object types” on page 195 for valid values.

z is &CHAROPT value. The default &CHAROPT value is ’1’.

The format of a general resource class name associated with DB2 administrative authorities is:

yyyyADMz

Where

yyyy is the DB2 subsystem or DB2 group attach name (from XAPLGPAT)

ADM is the class abbreviation for administrative authority

z is the &CHAROPT value. The default &CHAROPT value is ’1’.

The resource names do not contain the subsystem name.

With the single subsystem class scope, you can use the classes provided in the class descriptor table (CDT) supplied by IBM only if you are using the default DB2 subsystem name of ″DSN″ and have altered the &CHAROPT variable in the external security module to be a blank (’ ’). If this is not the case, you must define your own classes in the installation-defined class descriptor table. In addition, you need to define a separate set of classes for each subsystem that uses the RACF/DB2 external security module.

Multi-subsystem class scope (classification model II)

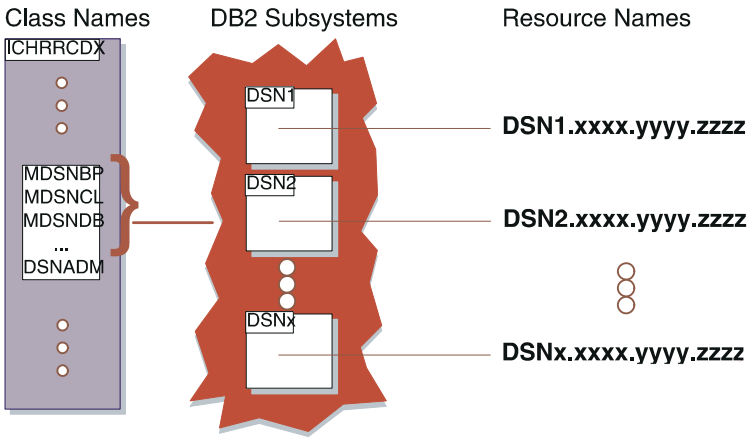


Figure 29. Multi-subsystem scope classes

When you select this model, the RACF/DB2 external security module places the DB2 subsystem name in the resource name. The resource names are prefixed with the subsystem name or the DB2 group attach name. The class names have the following format:

abbbbxxz

Where

b is M for member class or G or Grouping class

bbbb is the &CLASSNMT value. The default &CLASSNMT value is 'DSN'.

xx is the type of DB2 object (See "DB2 object types" on page 195 for valid values).

z is the &CHAROPT value. This is ignored if &CLASSNMT is 'DSN'.

The format of a general resource class name associated with DB2 administrative authorities for this scope is:

yyyyADMz

Where

yyyy &CLASSNMT value. The default &CLASSNMT value is 'DSN'.

ADM is the class abbreviation for administrative authority

z is the &CHAROPT value. This is ignored if &CLASSNMT is 'DSN'.

The resource names are prefixed with the subsystem name or DB2 group attach name.

If you use the multi-subsystem class scope and the default &CLASSNMT value ('DSN'), you can use the classes provided in the class descriptor table (CDT) supplied by IBM. All subsystems sharing the external security module can use the same set of classes.

You can change the &CLASSNMT if you do not want to use the default value. However, if you set &CLASSNMT to a value other than 'DSN', you need to define classes in the installation-defined class descriptor table.

Defining classes for the RACF/DB2 external security module (optional)

Attention

Defining classes for the RACF/DB2 external security module is optional. It is required *only* when you do not use the defaults.

When you change the &CLASSOPT or &CLASSNMT assembler SET symbols from their default values, you need to define classes in the installation-defined class descriptor table (CDT). You might also want to define corresponding entries in the RACF router table. For details on defining classes and router table entries, see "The class descriptor table (CDT)" on page 48 and "The RACF router table" on page 52.

It is not necessary to define classes for DB2 objects that are not protected by the external security module. The formats for these class names are defined in sections "Single subsystem class scope (classification model I)" on page 191 and "Multi-subsystem class scope (classification model II)" on page 193. To see which DB2 objects are protected, see *z/OS Security Server RACF Security Administrator's Guide*.

- When using the single subsystem class scope, the RACF/DB2 external security module creates class names dynamically by concatenating the DB2 group attach name for data sharing or the subsystem name with the object type. As a result, multiple DB2 subsystems can use the same copy of the RACF/DB2 external security module.

  Note: A set of classes must be defined for each subsystem.
- When using the multi-subsystem class scope, class names are created dynamically by concatenating this &CLASSNMT with the object type. As a result, any DB2 subsystem with the same &CLASSNMT can use the same copy of the RACF/DB2 external security module.

Note: If you choose to use installation-defined classes, you must use installation-defined classes with all objects for the same copy of the external security module. You cannot mix classes supplied by IBM and installation-defined classes. To use both types, you must use different versions of the external security module.

Installation-defined classes are expected to have the same class descriptor table attributes as the corresponding DB2 classes supplied by IBM.

DB2 object types

Following is a list of DB2 objects with the DB2 abbreviation used in the XAPL and the object abbreviation used for the RACF general resource class names (the *xx* value in the definition of class names):

Table 9. DB2 object abbreviations

DB2 Object	DB2 Object Abbreviation	RACF Class Abbreviation
Bufferpool	B	BP
Collection	C	CL
Database	D	DB
Java archive (JAR)	J	JR
Package	K	PK
Plan	P	PN
Schemas	M	SC
Storage Group	S	SG
Stored Procedures	O	SP
System	U	SM
Table/Index/View	T	TB
Tablespace	R	TS
User-Defined Distinct Types (UDT)	E	UT
User-Defined Functions (UDF)	F	UF
View	V	TB

Assembling and link-editing the RACF/DB2 external security module

To activate the RACF/DB2 external security module for a DB2 subsystem, you need to assemble the source and link-edit the module into the DB2 exit load library (*prefix*.SDSNEXIT). To do this:

1. Modify step 3 (JEX0003) of install job DSNTIJEX so it references the library containing the RACF/DB2 external security module for that subsystem.
2. Run this step.

You can use the RACF/DB2 external security module to protect multiple DB2 subsystems. When two or more subsystems use the same values as customization options, they can share the same copy of the RACF/DB2 external security module. To do this:

1. Modify step 3 (JEX0003) of install job DSNTIJEX to allow each DB2 subsystem to reference the same input library.
2. Run this step for each DB2 subsystem.

For more information on using DB2 install job DSNTIJEX, see *DB2 Installation Guide*.

After the RACF/DB2 external security module has been installed into the DB2 exit load library (*prefix*.SDSNEXIT), DB2 invokes it the next time the DB2 subsystem is started. If the initialization function is successful, the RACF/DB2 external security module is used for authority checking.

Attention

It is important *not* to install the RACF/DB2 external security module until the security administrator has defined and activated the desired classes and profiles.

RACF/DB2 external security module functions

DB2 calls the RACF/DB2 external security module during:

- DB2 initialization to create in-storage profiles
- DB2 authorization to check DB2 objects and authorities
- DB2 termination to delete in-storage profiles

The initialization function (XAPLFUNC = 1)

When the RACF/DB2 external security module is called with XAPLFUNC function code of 1, it issues a RACROUTE REQUEST=STAT request to determine if RACF is active. If RACF is not active, the RACF/DB2 external security module returns to DB2 with a return code of 12. If RACF is active, the RACF/DB2 external security module builds the class names, as specified by the assembler SET symbols, and performs a RACROUTE REQUEST=LIST,CLASS=*classname* for each new DB2-related class.

Attention

- If you override &CLASSNMT or use the single subsystem class scope, the RACF/DB2 external security module uses only installation-defined classes.
- If you use the multi-subsystem class scope with the default &CLASSNMT, the RACF/DB2 external security module uses classes supplied by IBM.

See Appendix A, “Supplied CDT classes,” on page 339 for a list of DB2 classes supplied by IBM.

The RACROUTE REQUEST=LIST,ENVIR=CREATE,GLOBAL=YES request brings profiles to a data space for that particular DB2 or allows a subsequent DB2 to use those in-storage profiles.

If no DB2-related classes were active, a failure occurs and the RACF/DB2 external security module ends with a return code of 12.

Note: The following are not failures:

- A class is not active (SAF RC=4, RACF RC=10)
- A class is not defined (SAF RC=4, RACF RC=8)

  If a class is not active or does not exist for an object or authority, the RACF/DB2 external security module defers to DB2 for authorization checking and ends with a return code of 4.

If *one* request fails, the *entire* initialization fails. When this happens, the RACF/DB2 external security module cleans up all the resources and ends with a return code of 12.

If you want to use DB2 classes for authorization against DB2 objects, the classes must be active when the subsystem is started.

Failures during initialization processing are indicated by a return and reason code pair and a message.

Initialization return and reason codes

Return Code (decimal)	Meaning	
0	Initialization successful	
	Reason Code (decimal)	**Meaning**
	0	Native DB2 authorization is used in the event of an error.
	16	The DB2 system is requested to stop in the event of an error on a subsequent authorization check.
12	Unable to service request; don't call exit again	
	Reason Code (decimal)	**Meaning**
	1	Input DB2 subsystem ACEE was not provided. Native DB2 authorization will be used.
	2	RACF is not active. Native DB2 authorization will be used.
	3	RACROUTE REQUEST=LIST,ENVIR=CREATE failure. Native DB2 authorization will be used.
	4	No active DB2 classes. Native DB2 authorization will be used.
	12	Input DB2 subsystem ACEE was not valid. Native DB2 authorization will be used.
	16	An initialization error occurred. The DB2 subsystem is requested to stop. The DB2 subsystem is requested to stop because DB2 Version 7 or later is installed and the installation option &ERROROPT was set to 2 in the RACF/DB2 external security module..

The authorization function (XAPLFUNC = 2)

The RACF/DB2 external security module performs FASTAUTH checks during authorization. Because there is no concept of negative access level in DB2, processing ends when the RACROUTE REQUEST=FASTAUTH request returns a return code of 0 or the list of checks for the request has been exhausted. For more information on the authorization checking function, see *z/OS Security Server RACF Security Administrator's Guide*.

Authorization checking return and reason codes

Return Code	Meaning

0 Access permitted

Reason Code (decimal)	Meaning
0	Access permitted by FASTAUTH checking.
13	Access permitted by implicit privilege of ownership.
14	Access permitted because current SQL ID matches schema name.

4 Unable to determine; perform DB2 authorization checking

Reason Code (decimal)	Meaning
0	Input class (XAPLTYPE) not active.
11	Input ACEE (XAPLACEE) not provided.
14	The ALET could not be created for cross memory ACEE.
15	Input privilege code (XAPLPRIV) or input class (XAPLTYPE) not defined to the RACF/DB2 external security module.
16	Input privilege code (XAPLPRIV) does not contain any rules.

8 Access denied

Reason Code (decimal)	Meaning
17	Autobind indicator (XAPLAUTO) is not zero indicating AUTOBIND was requested. Manual REBIND is required.

The termination function (XAPLFUNC = 3)

When the RACF/DB2 external security module uses XAPLFUNC function code 3, it issues a RACROUTE REQUEST=LIST,ENVIR=DELETE,GLOBAL=YES request. The classes that were previously brought into storage during DB2 initialization are deleted.

Failures during termination processing are indicated by a return and reason code pair and a message.

Termination return and reason codes

Return Code (decimal)	Meaning
0	Termination successful
8	Termination failure

Reason Code (decimal)	Meaning
1	Input DB2 subsystem ACEE was not provided.
7	RACROUTE REQUEST=LIST,ENVIR=DELETE failure.

12	Input DB2 subsystem ACEE was not valid.

Unsupported function codes

If the RACF/DB2 external security module receives a XAPLFUNC function code other than 1, 2 or 3, the RACF/DB2 external security module sends a return code of 12 to the caller.

When a return code of 12 is received:

- Native DB2 usthorization is used if &ERROROPT=1 or the level of DB2 is below DB2 Version 7.
- The DB2 subsystem stops if &ERROROPT=2 and the level of DB2 is DB2 Version 7 or later.

Chapter 7. RACF database utilities

RACF internal reorganization of aliases utility program (IRRIRA00) 203
IRRIRA00 stage conversion. 204
Diagnostic capability 206
Input for IRRIRA00 207
IRRIRA00 example 207
Output from IRRIRA00 207
RACF database initialization utility program (IRRMIN00) 208
Running IRRMIN00 when PARM=NEW is specified 209
Running IRRMIN00 when PARM=UPDATE is specified. 210
Running IRRMIN00 when PARM=ACTIVATE is specified 211
Diagnostic capability 211
Input for IRRMIN00 211
Output from IRRMIN00 212
RACF cross reference utility program (IRRUT100) 212
Diagnostic capability 214
The work data set 214
Using IRRUT100. 215
Input for IRRUT100. 215
Output from IRRUT100 216
RACF database verification utility program (IRRUT200) 218
Copying a data set in the RACF database 218
Diagnostic capability 219
Monitoring the capacity of the RACF database. 220
Processing considerations for databases from other systems 220
Using IRRUT200. 221
Input and output for IRRUT200 222
Utility control statements 223
Scanning the index blocks 224
Unformatted printout 225
Formatted printout 225
BAM/allocation comparison 228
IRRUT200 return codes 232
RACF database split/merge/extend utility program (IRRUT400). 232
How IRRUT400 works. 233
Using IRRUT400 to extend a database 233
Copying a RACF database 233
Repairing a RACF database 235
Diagnostic capability 236
Executing IRRUT400 236
Specifying the input database 237
Specifying the output database 237
Selecting the output data set 237
Processing the output data sets 238
Specifying parameters. 238
Processing of conflicts and inconsistencies 240
IRRUT400 return codes 241
IRRUT400 examples 242
Example 1. Copying a database 242
Example 2. Splitting a database 243
Example 3. Merging data sets 243
Example 4. Copying to a larger database. 243
Example 5. Unlocking a database 244
Example 6. Copying using a two—stage option 244

© Copyright IBM Corp. 1994, 2004

Utilities documented in other documents 245
RACF database unload utility program (IRRDBU00) 245
RACF remove ID utility (IRRRID00) 245
RACF SMF data unload utility program (IRRADU00) 245
BLKUPD command 245
Data security monitor (DSMON) 245
RACF report writer (RACFRW) 245
RRSF VSAM file browser (IRRBRW00) 246
RACFICE reporting tool 246

The RACF utilities are used for maintaining, modifying, copying, unloading, and monitoring the RACF database.

Table 10. RACF utilities described in this chapter

Utility	Description	More information
IRRIRA00	Converts an existing RACF database to use an alias index for application identity mapping	"RACF internal reorganization of aliases utility program (IRRIRA00)" on page 203
IRRMIN00	• Formats a non-VSAM DASD data set for use as a RACF database • Updates an existing RACF database with a new set of templates • Activates a new set of templates on a system	"RACF database initialization utility program (IRRMIN00)" on page 208
IRRUT100	Lists all the occurrences of a user ID or group name in the RACF database	"RACF cross reference utility program (IRRUT100)" on page 212
IRRUT200	• Provides information about the size and organization of a RACF database • Identifies inconsistencies in a RACF database • Copies a RACF database	"RACF database verification utility program (IRRUT200)" on page 218
IRRUT400	• Identifies inconsistencies in a RACF database • Copies a RACF database • Redistributes data between data sets in the RACF database • Reorganizes the RACF database	"RACF database split/merge/extend utility program (IRRUT400)" on page 232
Note: For a summary of RACF utilities described in other areas of the RACF library, see "Utilities documented in other documents" on page 245.		

Notes:

1. If you are sharing a database between MVS and VM, you should run the utilities from the MVS side for ease of recovery and error reporting.
2. If you are sharing a database, the templates must match the latest level of code. The IRRMIN00 utility (for the latest release) updates the database templates. The templates are downwardly compatible. For example, if OS/390 V2R6 and OS/390 V2R10 are sharing a database, the templates must be at the Release 10 level, but your release 6 system can successfully use the database if RACF is not enabled for sysplex communication. For additional considerations when RRSF is used, see "Shared RACF databases" on page 8.

3. Run OS/390 Security Server (RACF) utilities only on an OS/390 Security Server (RACF) system. Do not use Security Server utilities with an earlier release of RACF, and do not run utilities from an earlier release of RACF on your z/OS or OS/390 system. The exceptions to this are IRRMIN00 and IRRUT100, which can be run on a lower level system.
4. If you are sharing a RACF database between systems at different levels, you can run any of the utilities, except IRRMIN00, from any of the sharing systems. For example, if an OS/390 V2R10 system is sharing a database with an OS/390 V2R8 system, you can run IRRUT400 from either the Release 10 system or the release 8 system. See “RACF internal reorganization of aliases utility program (IRRIRA00)” for restrictions involving the IRRIRA00 utility.

RACF internal reorganization of aliases utility program (IRRIRA00)

This utility advances the application identity mapping stage for RACF databases created before OS/390 Release 10. You do not need to run the utility against databases created with IRRMIN00 PARM=NEW for OS/390 Release 10 or later because they are already initialized for the final stage of application identity mapping.

Application identity mapping in its final stage, stage 3, is an alternative to the use of mapping profiles to associate RACF user and group names with z/OS UNIX, Lotus® Notes®, and Novell Directory Services identifiers. For these associations, IRRIRA00 converts the database mapping profile information into an alias index, which uses less space. This conversion is accomplished through a series of stage transitions from an initial stage 0 to the completed conversion in stage 3. It is important to verify that your applications relying on the alias information continue to execute properly through the interim stages. Changes made to RACF user and group commands and callable services to support the alias indexes are intended to be transparent. However, you need to modify any application code that references or manipulates the mapping profiles directly to use the standard interfaces.

You can run the IRRIRA00 utility without specifying parameters to determine the current stage of the active RACF database. You cannot run the utility against an inactive database. The stage value is maintained in the ICB of the database master data set. If your database is split across multiple data sets, RACF assumes that they are at the same stage as the master.

IRRIRA00 updates all active data sets, both primary and backup, that make up the RACF database. All primary RACF data sets must be active to allow the utility to complete successfully.

- If the primary RACF data sets are active but the backup data sets are inactive, the utility updates only the primary data sets. A message is issued to indicate that the backup database was not changed.
- If some backup data sets are active and some are inactive, an error message is issued and processing ends without updating the primary database.

IRRIRA00 opens the master primary RACF data set and the master backup RACF data set, if it is active, to write the stage indicator into the ICB. You must have update authority to each data set to allow the data set to open successfully. Failing opens end with ABEND 913.

IRRIRA00 obtains serialization to prevent activities such as RVARY and SETROPTS commands from processing while the utility is running. Processing of

RACF commands that add, alter and delete user and group profiles might also be delayed. You should avoid RACF administration while the utility is running.

IRRIRA00 runs fastest when there is minimal activity on the system. For a database with a large number of mapping profiles, the utility converts from stage 0 to stage 1 in about half the time if you set the backup database inactive and run IRRIRA00 against the primary database only. You can use IRRUT200 or IRRUT400 to copy the primary database to the backup database after the utility completes successfully.

IRRIRA00 does not propagate the new alias index entries or the deleted mapping profiles to other databases with RRSF. You need to run the utility for each database when that system is ready to enter a new stage. RACF databases do not need to be at the same stage to be part of the same RRSF network unless specific code is used to manipulate mapping class profiles using RACROUTE or ICHEINTY. Command propagation works correctly between systems whose RACF databases are at different stages.

Attention

If RACF is enabled for sysplex communication, whenever you need to run IRRIRA00 against a database that is active on a system that is a member of the RACF data sharing group, always run the utility from a system in the group. If you do not, you might damage your RACF database, or receive unpredictable results from the utility.

IRRIRA00 stage conversion

To convert a database to use an alias index, you must run the IRRIRA00 utility to advance the database through a series of stages. You can perform a single transition for each IRRIRA00 invocation, moving from stage 0 to 1, 1 to 2, or 2 to 3. You cannot skip a stage or retreat to a previous stage. This multi-step approach is intended to give you manageable, uninterrupted use of your database through the conversion. You need to understand the stages and how they affect RACF.

Table 11. IRRIRA00 stage summary

Stage	Manager	Commands	Callable Services
0	• Does not maintain alias index • Purges VLF • Does not allow alias index entry locates	Maintains VLF and mapping profiles	Identity search order: 1. VLF 2. Mapping profile or database search
1	• Maintains alias index • Purges VLF • Does not allow alias index entry locates	Maintains VLF and mapping profiles	Identity search order: 1. VLF 2. Mapping profile or database search
2	• Maintains alias index • Purges VLF • Allows alias index entry locates	Maintains VLF and mapping profiles	Identity search order: 1. Alias index entry locate 2. VLF 3. Mapping profile or database search
3	• Maintains alias index • Allows alias index entry locates	Does not maintain VLF and mapping profiles	Identity search order: 1. Alias index entry locate

Table 11. IRRIRA00 stage summary (continued)

Stage	Manager	Commands	Callable Services
Notes: 1. Mapping profiles are used if the appropriate class is active (for example, UNIXMAP, NOTELINK, NDSLINK). If UNIXMAP is not active, RACF searches through all the user and group profiles in the database with an OMVS segment until a match is found for the GID or UID. 2. VLF is applicable only for an OMVS UID or GID. The IRRUMAP or IRRGMAP class must be active.			

Before you begin

- Before advancing the stage of your database, make a copy of the database for recovery purposes.
- If you are sharing a database with a lower level system, review "Using Application Identity Mapping" on page 83 before proceeding with the conversion.

Stage 0

The database does not have an alias index and the RACF database manager does not attempt to use or maintain the alias index. It continues to use the mapping profiles. Any database created earlier than OS/390 Release 10 exists in stage 0 automatically until you convert it with IRRIRA00.

Stage 1

Before advancing to stage 1

Before entering this stage, run IRRMIN00 PARM=UPDATE and IPL the system if it was not done during previous migration steps.

To enter stage 1, IRRIRA00 sets the stage indicator in the ICBs and the RCVT to 1 and builds the alias index based on the information in the RACF database.

In stage 1, the database contains the existing mapping profiles and the new alias index. RACF uses VLF and the mapping profiles to locate a base USER or GROUP profile name that has been given another product's identity information. The RACF database manager maintains an alias index but does not use it to locate user or group names. RACF user and group commands such as ADDUSER maintain both the mapping profiles and the alias index entries during addition, modification, or deletion of USER and GROUP profiles.

Stage 2

To enter stage 2, IRRIRA00 sets the stage indicator in the ICBs and RCVT to 2.

In stage 2, RACF maintains both alias index entries and mapping profiles. The RACF database manager can use the alias index to locate user and group names.

At this stage, the identity mapping callable services look up application IDs in an alias index to retrieve corresponding RACF user or group names. If the entry is not found in the index, RACF searches through VLF, mapping

profiles, or base profiles, depending on the alias type and active classes. If the secondary search locates the alias index entry successfully, the callable services:

- Return the associated user or group name
- Write a LOGREC entry indicating that the alias index does not match other mapping information

The callable services also generate a LOGREC entry if the search fails for any reason other than `not found`, regardless of the success of a secondary search.

To identify the error types, look for LOGREC entries that include the string `IRRRUM01`, `IRRRGM01`, or `IRRRIM00` in the ″FREE FORMAT COMPONENT INFORMATION″ section. See *z/OS Security Server RACF Callable Services* to help you interpret return codes and determine how to correct any errors. You must resolve the problems before moving to stage 3.

Stage 3

Before advancing to stage 3

- You can advance to stage 3 after successfully operating in stage 2. Before entering stage 3, check the LOGREC entries and correct any errors that might have occurred when the callable services searched for alias index entries during stage 2.
- You should enter stage 3 only when all sharing systems have the OS/390 Release 10 Security Server or later installed. If you are sharing a RACF database with a system that is at a lower level, you might receive unpredictable results.

To enter stage 3, IRRIRA00 sets the stage indicators to 3 in the ICBs and the RCVT and deletes the mapping profiles from the database.

In stage 3, RACF uses only alias index entries, not mapping profiles, for UID, GID, SNAME, and UNAME associations. Commands such as ADDUSER no longer maintain the old mapping profiles. Entries are not placed in VLF, which is no longer used to locate profiles. You can remove the IRRUMAP and IRRGMAP VLF classes from the COFVLFxx member of the parmlib. You can also deactivate the RACF UNIXMAP, NOTELINK, and NDSLINK classes.

A database created in OS/390 Release 10 or later is automatically set to stage 3.

Diagnostic capability

IRRIRA00 does not provide RACF database diagnostic information. If you suspect a RACF database error, you should start your problem determination by running the IRRUT200 utility and requesting the INDEX and MAP ALL functions. For details, see “RACF database verification utility program (IRRUT200)” on page 218.

See Chapter 9, “Recovery Procedures,” on page 303 and *z/OS Security Server RACF Diagnosis Guide* for more information on RACF database diagnosis and correction.

Input for IRRIRA00

IRRIRA00 expects the following parameter on the JCL EXEC statement:

PARM=STAGE(*n*), with *n*=1,2,3 to specify the desired level of the system. The utility:

1. Checks the current level of the system to be sure it is at the *n*-1 level.
2. Performs the necessary actions to enable the specified state *n*.

If no parameter is specified, the current stage is listed.

Job control statements for IRRIRA00 require the following data definition:

ddname	Description
SYSPRINT	Defines the output data set for processing and error messages.

IRRIRA00 example

In this example, IRRIRA00 converts an existing RACF database from stage 1 to stage 2 for application identity mapping function:

```
//DBSTAGE   JOB
//STEP      EXEC   PGM=IRRIRA00,PARM=STAGE(2)
//SYSPRINT  DD     SYSOUT=A
```

Output from IRRIRA00

A return code greater than 4 indicates that the stage conversion did not complete successfully. If appropriate, correct the errors indicated by the messages and run the utility again. IRRIRA00 issues no message when the return code is x'14' (20 decimal) because SYSPRINT cannot be opened to write the message. In this case, you should verify that the SYSPRINT DD statement is correct and that the utility can access the specified file.

The IRRIRA00 program sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.
4	(4)	One of the following warning messages is issued: • Database already at requested stage. • Backup database not converted, currently inactive.
C	(12)	Terminating error. I/O error reading or writing the ICB.
10	(16)	Terminating error. One of the following occurred: • RACF is not active • Cannot establish recovery • Parameter error - unsupported stage value • Parameter error - unrecognized keyword • Parameter error - not permitted to convert from current stage to stage value specified • Failure reading/updating profile • Conversion cannot be performed because system is in read-only mode • Failure writing to CF • Conversion cannot be performed because templates are downlevel
14	(20)	Terminating error. Unable to open SYSPRINT.
20	(32)	Terminating error. RACF not enabled.

RACF database initialization utility program (IRRMIN00)

This utility initializes a RACF database, and updates the database copy and the in-storage copy of the database templates. You can use it in three ways:

- Use PARM=NEW to initialize a new, empty database.
- Use PARM=UPDATE to update an existing database with a new set of RACF templates.
- Use PARM=ACTIVATE to replace the in-storage templates with a new set of RACF templates.

For information on templates, including information about how to apply them to your system when they are updated by a new release or PTF, see “Database templates” on page 5.

If you have split your database, you must run IRRMIN00 against each data set defined in your data set name table (ICHRDSNT). If you have a backup database, you must also run IRRMIN00 against each data set in the backup database.

You can use the SET LIST command to display the level of the templates that your system is using. The level is an FMID, such as HRF7708, or an APAR number, followed by an 8-digit representation of the release level and an 8-digit representation of the APAR level. For example:

```
RACF STATUS INFORMATION:
         TEMPLATE VERSION          - HRF7708 00000020.00000010
         DYNAMIC PARSE VERSION     - HRF7708
```

When comparing templates to determine which is the most recent, RACF first compares the 8-digit representations of their release levels. The templates having the highest release level are considered to be the latest. If the release levels are the same, RACF compares the 8-digit representations of the APAR levels, and the templates having the highest APAR level are considered to be the latest. For templates earlier than FMID HRF7708, which do not have 8-digit representations of the release level and APAR level, the release level and APAR level are each assumed to be 00000000.

If you install a new release of RACF or a PTF that requires a re-IPL and contains an update to the RACF templates (shipped in CSECT IRRTEMP2), you should first run the latest version of IRRMIN00 with PARM=UPDATE to write the templates from IRRTEMP2 to the RACF database. Then do the required re-IPL. During the IPL, RACF initialization builds the in-storage templates from the updated database templates. If you were installing a new release, remember to include a steplib to the new SYS1.LINKLIB in your JCL for IRRMIN00 PARM=UPDATE.

Note: If you do not run IRRMIN00 to update your database before you re-IPL, RACF initialization determines that the database does not have the latest level of the templates, ignores the templates in the database, and automatically uses the latest templates shipped in the CSECT IRRTEMP2. However, until you run IRRMIN00 you might get error messages from IRRUT200 or BLKUPD during some operations, and the RACF database unload utility will not unload new fields. Also products that read the database directly and process the database template blocks will have problems with profile information related to the new templates.

If you install a PTF that contains an update to the RACF templates but does not require a re-IPL (because all the modules in the PTF reside in LINKLIB), first run IRRMIN00 with PARM=UPDATE to update the database templates. Then run

IRRMIN00 with PARM=ACTIVATE to have RACF replace the in-storage templates with the database templates. An IPL is not required.

You do not have to enable RACF in order to run IRRMIN00 with PARM=NEW or PARM=UPDATE.

Attention

- If RACF is enabled for sysplex communication, whenever you need to run IRRMIN00 against a database that is active on a system that is a member of the RACF data sharing group, always run the utility from a system in the group. If you do not, you might damage your RACF database, or receive unpredictable results from the utility.
- When IRRMIN00 JCL includes a STEPLIB other than SYS1.LINKLIB, it must be an APF-authorized library.

The ADDCREATOR and NOADDCREATOR keywords on the SETROPTS command determine whether RACF adds the user ID that creates a profile to the access list for the profile. The initial setting of these keywords depends on whether your database is new or old. If you run IRRMIN00 with PARM=NEW, the initial setting is NOADDCREATOR. If you run IRRMIN00 with anything other than PARM=NEW, RACF retains the current value of ADDCREATOR or NOADDCREATOR. For compatibility and migration reasons, ADDCREATOR is the default if no prior specification of ADDCREATOR or NOADDCREATOR has occurred. For more information on the ADDCREATOR and NOADDCREATOR keywords on the SETROPTS command, see *z/OS Security Server RACF Command Language Reference*.

Running IRRMIN00 when PARM=NEW is specified

When you specify PARM=NEW, the RACF database initialization program (IRRMIN00) formats a non-VSAM DASD data set so that it can be used as a RACF database.

You must run IRRMIN00 with PARM=NEW during the initial installation of RACF to format the RACF database. After RACF is installed, you can run IRRMIN00 with PARM=NEW to format an alternate RACF database.

If you attempt to run IRRMIN00 with PARM=NEW for a RACF data set that is active on the system from which you are running the utility, IRRMIN00 issues an error message and ends. This behavior prevents you from overwriting a RACF data set that is active on the system. It does not, however, prevent you from overwriting a RACF data set that is inactive on the system from which you are running, but active on another system.

Attention

Do not run IRRMIN00 PARM=NEW against an existing RACF database unless you do not need the data in that database. PARM=NEW processing destroys all existing data as it formats an empty database for you.

When your system is at OS/390 Release 10 or higher, IRRMIN00 PARM=NEW causes RACF to set the application identity mapping stage indicator to stage 3 in the ICB of the new data set. If you share a new database with one or more systems at a lower release level, you need to be sure the database can operate in stage 0. To accomplish this, be sure to run IRRMIN00 PARM=NEW from the lower level system first. When you're done, you should run IRRMIN00 PARM=UPDATE from the higher level system. These actions set the stage indicator to 0 and ensure consistent use of mapping profiles among the sharing systems. For more information on using application identity mapping in a sysplex, see "Using Application Identity Mapping" on page 83.

When formatting a database with PARM=NEW, the database must not be active on any system.

The IRRMIN00 program divides a RACF database into 4K records. When you create a new RACF database, the following records are initialized:

Record	Description
ICB	The header block (inventory control block).
Templates	The user, group, data-set, and general template definitions, the alias-related template extension, plus five reserved blocks.
Segment table block	Segment definitions from within a template.
BAM blocks	BAM (block availability mask) blocks are initialized with the space configuration for the database.
Empty blocks	Available for later use as profile blocks or index blocks.

Note: No profile or index blocks are initialized.

Running IRRMIN00 when PARM=UPDATE is specified

When you update a RACF database, IRRMIN00 adds the new templates to the database, writes the segment table, and updates the pointers and counts in the ICB to reflect the new templates. The utility does not alter the index blocks and profiles, or update the in-storage copy of the templates that RACF uses.

When you specify PARM=UPDATE, IRRMIN00 compares the level of the templates in the RACF database (which it detemines from the ICB) with the level of the templates in the CSECT IRRTEMP2. If IRRTEMP2 has a template level less than or equal to that in the ICB, IRRMIN00 issues a message to SYSPRINT and does not update the database. This behavior prevents you from accidentally installing a down-level version of the templates.

If you are updating the active RACF database, IRRMIN00 obtains an exclusive RESERVE on the database. If RACF is running in data sharing mode, an ENQ is issued instead of the RESERVE. The RESERVE or ENQ should be of short

duration and should not interfere with other work active on the system. In read-only mode, RACF does not allow IRRMIN00 to be run against an active database.

Running IRRMIN00 when PARM=ACTIVATE is specified

When you run IRRMIN00 specifying PARM=ACTIVATE, IRRMIN00 compares the template level of the RACF database (which it determines from the ICB of the master primary RACF data set) to the level of the templates being used by the system. If the level of the templates on the RACF database is higher, IRRMIN00 makes an in-storage copy of the templates from the database, thus activating them on the system. The utility issues a message indicating the new template level when it begins template activation, and issues another message when it complates template activation.

If any of the following conditions are true, IRRMIN00 issues an error message and does not activate a new set of templates:

- The level of RACF on the system does not support template activation.
- RACF is not active.
- The primary master RACF data set is not active.
- The level of the templates that are active on the system is higher than or equal to the level of the templates on the RACF database.

Diagnostic capability

IRRMIN00 does not provide RACF database diagnostic information. If you suspect a RACF database error, you should start your problem determination by running the IRRUT200 utility and requesting the INDEX and MAP ALL functions. For details, see “RACF database verification utility program (IRRUT200)” on page 218.

See Chapter 9, “Recovery Procedures,” on page 303 and *z/OS Security Server RACF Diagnosis Guide* for more information on RACF database diagnosis and correction.

Input for IRRMIN00

IRRMIN00 expects one of the following parameters on the JCL EXEC statement:

- PARM=NEW, specified when formatting a new RACF database
- PARM=UPDATE, specified when updating the templates on an existing database. IRRMIN00 adds or updates the templates as required and writes the segment table, while leaving the old profiles intact. PARM=UPDATE is the default if no parameter is specified.
- PARM=ACTIVATE, specified when activating a new version of the templates.

IRRMIN00 requires the following DD statements:

ddname	Description
SYSPRINT	Defines the output data set. The minimum block size is 129 (enforced by a DCB exit).
SYSRACF	Defines a contiguous, unmovable, non-VSAM data set to be formatted. The logical-record size and block size are required to be 4096. For PARM=NEW, this is forced by RACF utility processing. If you are updating an existing RACF data set, IRRMIN00 updates the RACF data set in place.

Note: Before z/OS Version 1 Release 5, IRRMIN00 required a SYSTEMP DD statement. If you have a SYSTEMP DD statement in existing IRRMIN00 JCL, and you do not remove it, it will be ignored.

For an example of the JCL required to allocate space, catalog the data set, and run the IRRMIN00 program, see "Creating a RACF database" on page 10.

Output from IRRMIN00

RACF writes the input images from the template definitions to the printer along with messages indicating errors or success.

The IRRMIN00 program sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.
4	(4)	Attention—the RACF database is usable, but the target of the SYSRACF DD statement might be wrong, or the template level on the RACF database or the level of IRRMIN00 executed might not be the one expected.
C	(12)	The program encountered a terminating error. • For PARM=NEW or PARM=UPDATE, the RACF database was not formatted or reformatted. • For PARM=ACTIVATE, the templates from the database were not activated.
10	(16)	The output database could not be opened. The RACF database was not formatted.
14	(20)	The program was entered at an incorrect entry point.

RACF cross reference utility program (IRRUT100)

IRRUT100 is a RACF utility program that lists all occurrences of a user ID or group name that are in a RACF database. It uses the RACF manager to access the RACF database and locate possible occurrences of a user ID or group name.

An alternative to using IRRUT100 is to use the database unload utility, IRRDBU00. It provides a sequential file of the database that an installation can manipulate to obtain additional and more complex reports. For more information on IRRDBU00, see *z/OS Security Server RACF Macros and Interfaces* and *z/OS Security Server RACF Security Administrator's Guide*.

To invoke IRRUT100, you must be a RACF-defined user and either have the SPECIAL, group-SPECIAL, AUDITOR, or group-AUDITOR attributes, or be requesting a list of occurrences for only your user ID.

Although IRRUT100 must read every user and group profile in your database, it obtains and releases database serialization for each profile being read. Thus, the database *is* accessible, depending on the performance options set at your installation and other ongoing system activity.

IRRUT100 produces a cross-reference report that describes the occurrences of each user ID or group name specified. The letter G in parentheses follows each generic profile name.

For groups, IRRUT100 provides information on the following occurrences:

- The group name exists in the RACF database.
- The group is a subgroup of group *xx*.
- The group is a superior group of group *xx*.
- The group is the default group for user *xx*.
- The group is a connect group for user *xx*.
- The group was the connect group when the user created data set profile *xx*.
- The group name is the high-level qualifier of data set profile *xx*.
- The group has standard access to data set profile *xx*.
- The group has standard access to general resource *xx*.
- The group is the owner of user *xx*.
- The group is the owner of group *xx*.
- The group is the owner of data set profile *xx*.
- The group is the owner of general resource *xx*.
- The group is the owner of connect profile *xx*.
- The group exists in the conditional access list of general resource profile *xx*.
- The group exists in the conditional access list for data set profile *xx*.
- The group is the resource owner of profile *xx*.
- The group is a member of the GROUPS field in the TME® segment of ROLE-class general resource profile *xx*.

For user IDs, IRRUT100 provides information on the following occurrences:

- The user ID exists in the RACF database.
- The user is the owner of group *xx*.
- The user is listed as a member of group *xx*. The user might not be listed as a member of the group if it is a universal group.
- The user is the owner of user *xx*.
- The user is the owner of data set profile *xx*.
- The user is the owner of general resource *xx*.
- The user has standard access to data set profile *xx*.
- The user has standard access to general resource *xx*.
- The user ID is the high-level qualifier of data set profile *xx*.
- The user is the owner of connect profile *xx*.
- The user is to be notified when access violations occur against data set *xx*.
- The user is to be notified when access violations occur against general resource *xx*.
- The user exists in the conditional access list of data set profile *xx*.
- The user exists in the conditional access list of general resource profile *xx*.
- The user is the resource owner of profile *xx*.
- The user appears as RACLINK entry (user ID association) *node.userid* for user ID profile *xx*.
- The user ID is the second qualifier of FILE profile *xx*.
- The user ID is the second qualifier of DIRECTRY profile *xx*.
- The user exists in the application data field of general resource profile *xx*.

See Figure 30 on page 217 for a sample output of the printed report that IRRUT100 produces.

Exit Routine

RACF provides a preprocessing exit for an installation-written routine when the IRRUT100 utility is invoked. For more information, see “ICHCNX00 processing” on page 262.

Diagnostic capability

IRRUT100 is not designed to provide RACF database diagnostic information. It does, however, read many of the profiles in the database and in so doing might (implicitly) identify profiles with errors. If you suspect a RACF database error, you should start your problem determination by running the IRRUT200 utility and requesting the INDEX and MAP ALL functions. For details, see “RACF database verification utility program (IRRUT200)” on page 218.

See Chapter 9, “Recovery Procedures,” on page 303 and *z/OS Security Server RACF Diagnosis Guide* for more information on RACF database diagnosis and correction.

The work data set

Records in the work data set are 261 bytes long, keyed, and unblocked. Each record is formatted as follows:

Bytes Description

Bytes 0-2:
Relative block address of the next record on the chain. A relative block address of 0 indicates the end of the chain. Each input name has one chain.

Byte 3:
Record-type code, which corresponds to a SYSOUT message as follows:

- **X'01'** Beginning of the chain for this input name
- **X'02'** Group name exists. (Name is blank.)
- **X'03'** In the subgroup list of group name
- **X'04'** Superior group of group name
- **X'05'** Owner of group name
- **X'06'** In the access list of group name
- **X'07'** User entry exists. (Name is blank.)
- **X'08'** Owner of user name
- **X'09'** Default group for user name
- **X'0A'** Connect group for user name
- **X'0B'** First qualifier of data-set profile name or qualifier supplied by an exit routine
- **X'0C'** Owner of data-set profile name
- **X'0D'** In the standard access list of data-set profile name
- **X'0E'** Create group of data-set profile name
- **X'0F'** Owner of resource name
- **X'10'** In the standard access list of the general-resource profile
- **X'11'** Owner of the connect profile name
- **X'12'** In the notify field of the data-set profile
- **X'13'** In the notify field of the general-resource profile
- **X'14'** In conditional access list of the data-set profile
- **X'15'** In conditional access list of the general-resource profile
- **X'16'** Resource owner of profile
- **X'17'** Appears as RACLINK entry (user ID association) *node.userid* for the user ID profile
- **X'18'** Qualifier of the general resource profile. (This is used only for FILE and DIRECTRY profiles.)
- **X'19'** Member of GROUPS field in TME segment of general resource profile
- **X'20'** In application data field of general resource profile

Byte 4-5:
Length of entry name

Bytes 6-260:
User name, group name, data-set profile name, connect profile name, or the class name, followed by the resource name. (These names are associated with the record type indicated in byte 3.)

All of the type 1 records are located at the beginning of the data set. The name field for the type 1 records is the input name. The records for the occurrences of the input name are chained to this record by the relative block addresses.

Using IRRUT100

IRRUT100 uses a control data set as input. The control data set contains the utility control statements that indicate the names to cross-reference.

IRRUT100 produces the following output:

- A message data set containing the results of the IRRUT100 operations. The message data set includes the printed report and any error messages.
- A work data set containing the internal records describing the occurrences of each input name. This work file provides the data for the listed report and can be kept at the end of the job for other applications. If you request the records for a universal group, you should be prepared to provide enough space for a very large work data set.

Input for IRRUT100

IRRUT100 is controlled by job control statements and utility-control statements. The job control statements are necessary to execute or invoke the program and to define the data sets used and produced by the program. The utility control statements specify the names to be cross-referenced.

Job control statements: The following job control statements are necessary for using IRRUT100.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IRRUT100) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to an output device, a tape volume, or a direct-access device.
SYSUT1 DD	Defines a work data set on a direct-access device.
SYSIN DD	Defines the control data set. The control data set is normally found in the input stream; however, it can be a member of a procedure library or a sequential data set existing elsewhere.

Note: If the utility is executed under TSO, you can allocate both the SYSIN and SYSPRINT data sets to the terminal.

The format of IRRUT100 SYSIN is

```
name  [name]

/END
```

where:

name is a group name or user ID that is one to eight characters long and begins in any column.

Names are separated either by commas or blanks.

You can use only columns 1 through 72; continuation characters are not allowed. If all the names do not fit on one statement, you can use additional statements of the same format. The maximum number of names you can specify is 1000.

/END ends the utility program. The /END statement is not required. If it is coded, this statement must begin in column 1. An end-of-file on the SYSIN data set also terminates the program.

IRRUT100 example: In this example, IRRUT100 locates all occurrences of the group name RACG0001 and the user ID RACU002 in the RACF database and prints these occurrences on the system output device.

```
//XREF      JOB
//STEP      EXEC   PGM=IRRUT100
//SYSUT1    DD     UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
 RACG0001   RACU002
/END
```

Output from IRRUT100

Figure 30 on page 217 shows an example of output from IRRUT100.

```
1
 Occurrences of GROUPMID

 Owner of DASDVOL  DOWNER
 In standard access list of general resource profile DASDVOL  DGROUP
 Create group of profile USERMID.GROUP.TEST (G)
 Owner of profile HILDE.OWNER.DATASET
 First qualifier of profile GROUPMID.SAMPLE.DATASET
 In standard access list of data set profile GROUPLOW.ACCESS.DATASET
 Owner of connect profile USERMID2/SYS1
 Owner of connect profile USERMID1/SYS1
 Owner of group GROUPOWN
 Superior group of group GROUPOWN
 Group name exists
 Superior group of group GROUPLOW
 In subgroup list of group GROUPHI
 Connect group for user USER3
 Connect group for user USER2
 Connect group for user USER1
 Connect group for user USERMID1
 Owner of user USERMID1
 Connect group for user USERMID
 Default group for user USERMID
 Connect group for user HILDE

 (G) - Entity name is generic.
1
1
 Occurrences of USER1

 In notify field of general resource profile DASDVOL  DUSER1
 In conditional access list of general resource profile DASDVOL  DUSER1
 Owner of profile USER2.OWN.DATASET
 Owner of profile USER1.SAMPLE.DATASET
 First qualifier of profile USER1.SAMPLE.DATASET
 In standard access list of data set profile USERMID.GROUP.TEST (G)
 In notify field of data set profile HILDE.NOTIFY.CNTL
 In conditional access list of data set profile HILDE.COND.ACCESS
 Owner of connect profile USER2/SYS1
 Owner of connect profile USER2/GROUPMID
 Owner of group UGRP1
 In access list of group SYS1
 In access list of group GROUPMID
 RACLINK entry is present in user profile USER2 as MVS1.USER1
 Owner of user USER2
 User entry exists
 RACLINK entry is present in user profile SIVLE as MVS2.USER1
 Qualifier of general resource profile FILE FP1.USER1.DIR1.MIKAELA.MEM
 Qualifier of general resource profile DIRECTRY FP1.USER1.**  (G)

 (G) - Entity name is generic.
1
```

Figure 30. Sample output from IRRUT100

The IRRUT100 utility sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.

Hex	(Decimal)	Meaning
20	(32)	IRRUT100 has ended because the RACF product is not enabled. For information on enabling RACF, see “Enabling and disabling RACF” on page 62. For information on enabling products, see *z/OS MVS Product Management*. After you make the updates required to enable RACF, you must re-IPL in order for the updates to take effect.

RACF database verification utility program (IRRUT200)

IRRUT200 is a RACF utility program that you can use to identify inconsistencies in the internal organization of each data set comprising a RACF database and to make an exact copy of a RACF data set. You can also use it to monitor the usable space in a data set. It performs the following functions:

- Validates and reports errors found in the relative byte addresses (RBAs) of each segment of all profiles.
- Validates that index entries point to the correct profile.
- Validates the data set format.
- Issues return codes to signal validation errors.
- Scans the index blocks and prints information about problems with the index-block chains.
- Compares the segments of the data set that are actually in use to the segments allocated according to the BAM blocks, and prints information about inconsistencies.
- Processes the alias index to detect errors and display index blocks.
- Creates an encoded map for each BAM block in the RACF data set, which can be used to determine the amount of space left in the data set and how fragmented that space is. You can use this information to decide if the data set needs to be enlarged, or if it needs to be rebuilt in order to undo the fragmentation that has occurred over time.
- Creates a backup copy of a RACF data set. You can use this function if the backup data set has become out of synchronization with the primary data set.
- Creates an enhanced, formatted index report displaying the 255-byte profile name and profile type information.

Attention

If RACF is enabled for sysplex communication, whenever you need to run IRRUT200 against a data set that is active on a system that is a member of the RACF data sharing group, always run the utility from a system in the group. Failure to do so can cause the utility to build an incorrect output data set, or can cause erroneous results in the reports generated during the verification phase.

Copying a data set in the RACF database

IRRUT200 serializes on the input RACF data set and creates an exact, block-by-block copy of it. This exact copy can help performance when you are maintaining statistics on your backup data set. You can also use it to synchronize a backup data set with a primary data set. After IRRUT200 finishes the copy, serialization is released on the data set. If you plan to use the copy, you should

ensure that no updates occur between the conclusion of the copying and the time that RACF starts using the copy, or information could be lost.

IRRUT200 copies the RACF data set using the MVS utility IEBGENER. Anyone using IRRUT200 must have sufficient authorization for IRRUT200 and IEBGENER if these programs are protected by RACF. Copy performance can be improved by:

- Adjusting the BUFNO option on the SYSRACF DD statement and SYSUT1 DD statement
- Using DFSORT's ICEGENER (or an equivalent product) as a replacement for IEBGENER

Refer to *DSFMS/MVS Utilities* for tuning information on IEBGENER. Refer to *z/OS DFSORT Installation and Customization* for information on installing ICEGENER as a replacement for IEBGENER.

You can use IRRUT200 only if you are creating a copy of the data set that is the same size and on the same device type as the input data set. By same device type, we mean the track geometry must be the same (for example, you can copy between a 3390 Mod 2 and a 3390 Mod 3, but not between a 3390 and a 3380). To change the data set size or to copy a data set to a different device type, use IRRUT400. To copy a data set to tape, use a two-step process:

1. Use IRRUT200 to create a backup copy of the data set on disk.
2. Use another utility (for example, IEBGENER) to copy the backup copy to tape.

The target of the copy should *not* be an active RACF data set. If you need to refresh an active RACF data set, use RVARY to deactivate the data set before running IRRUT200. After utility processing completes, use RVARY to activate the data set.

To avoid copying a data set over itself, be sure that you specify different data set names for SYSRACF and SYSUT1.

Diagnostic capability

IRRUT200 is designed to detect errors in the internal organization of the RACF database. If you suspect a RACF database error, start your problem determination by running this utility requesting the INDEX and MAP ALL functions. If your database has more than one data set, run the utility against each data set that you suspect might be in error. When the job completes, inspect the utility return code. If the return code is zero, it is likely that the data set is okay, but some errors could still exist (see the additional diagnostic information below). If the return code is nonzero, review the output produced by the utility. Most often, a search for "IRR62" messages brings you quickly to the reported error.

See Chapter 9, "Recovery Procedures," on page 303 and *z/OS Security Server RACF Diagnosis Guide* for more information on RACF database diagnosis and correction.

Additional diagnostic information:

1. IRRUT200 checks most of the internal organization of a RACF data set, concentrating on the index structure; however, it does not verify every field within a profile. Therefore, it is possible for IRRUT200 to run and produce a zero return code even though the RACF data set contains a profile in error.

If you suspect that your data set contains such an error, we recommend that you run the RACF database unload utility, (IRRDBU00). The IRRDBU00 utility must read every profile in the database and thereby might (implicitly) identify profiles with errors.

For more information, see the IRRDBU00 description in *z/OS Security Server RACF Security Administrator's Guide*.

2. If IRRUT200 reports errors on upper-level index blocks only—that is, all profile blocks and level 1 (sequence set) blocks are okay—then you can use the IRRUT400 utility to create a new copy of the RACF data set. This works because the IRRUT400 utility does not use the upper-level index blocks. In fact, it reads only the sequence set blocks from the input data set and builds new upper-level blocks on the output data set. Therefore, your upper-level index block problems might be eliminated by using IRRUT400 to create a new RACF data set.

Monitoring the capacity of the RACF database

You can use IRRUT200 to monitor the capacity of the RACF database. Run IRRUT200 periodically against each data set in the database in order to determine if the data set is about to run out of space. If you have already run out of space (profile updates have failed due to an "insufficient space" condition), you can use IRRUT200 to determine if you need to enlarge the data set, or to rebuild it to undo any fragmentation that has occurred.

To monitor the database or diagnose an "insufficient space" condition, use the MAP ALL function. In addition to detecting BAM allocation inconsistencies, the MAP ALL function reports on the amount of space in use in the data set and produces an encoded map of the BAM blocks that you can use to determine if significant fragmentation has occurred. Either case can result in a profile create or update failing because no contiguous slot large enough to contain the new or changed profile is available. Once you determine that a data set is in danger of running out of space, or an "insufficient space" condition has been reached, use IRRUT400 to copy the data set to a new one. The new data set can be larger if you require more space, or it can be the same size if fragmentation is the only problem, because IRRUT400 rebuilds the data set while copying it, undoing any fragmentation that has occurred.

Processing considerations for databases from other systems

For proper utility operation, the enhanced-generic-naming (EGN) setting of the database that you are processing with IRRUT200 should be the same as the EGN setting of the system on which the utility is being executed.

To determine whether this affects you, answer these two questions:

- Are you processing a live (primary or backup) data set? If you are, you need not worry about the EGN setting.

  You can use the RVARY LIST command to see the RACF data sets that are currently in use.

- Is the EGN setting of the other database you are processing the same as the system EGN setting?

  If it is, you need not worry about the EGN setting. To find the EGN setting of the current system, you can issue the SETR LIST command. To find the EGN setting of the database:

- Issue the BLKUPD command against the first data set in the database that you are processing. Look in the data set name table (ICHRDSNT) to determine the name of the data set.
- Read record zero by issuing the READ X'00' BLKUPD subcommand.
- List the 195th byte by issuing the LIST RANGE(194,1) BLKUPD command. If the listed value has the low-order bit on, EGN is enabled. If the bit is off, EGN is not enabled.

Using the IRRUT200 utility in a mixed EGN environment might cause a question mark (?) to be displayed as a part of a formatted index entry.

Using IRRUT200

Attention

If RACF is enabled for sysplex communication, whenever you need to run IRRUT200 against a data set that is active on a system that is a member of the RACF data sharing group, always run the utility from a system in the group. Failure to do so can cause the utility to build an incorrect output data set, or can cause erroneous results in the reports generated during the verification phase.

RACF sysplex data sharing

The following discussion about running the IRRUT200 utility refers to the use of RESERVE to serialize access to a RACF data set while the utility is processing. If RACF is enabled for sysplex communication and is operating in data sharing or read-only mode, RACF uses ENQ instead of RESERVE.

The three ways to run IRRUT200 follow:

1. If you specify a work data set on the SYSUT1 DD statement, IRRUT200 RESERVEs the RACF data set specified on the SYSRACF statement and copies it to the work data set. After IRRUT200 has copied the data set specified on the SYSRACF statement from the RACF database to the work data set, IRRUT200 releases the RESERVE on the RACF data set.

   At no time is there a RESERVE on the work data set.

   IRRUT200 then uses the copy to find inconsistencies, and creates a printout identifying them.

   The space you specify on your SYSUT1 DD statement must be the same size as that of your RACF data set.

   Note: This is the recommended method, because it serializes against the RACF data set only during the copy phase, which is much shorter than the verification phase.
2. If you do not specify a work data set on the SYSUT1 DD statement, IRRUT200 reserves the RACF data set specified on the SYSRACF statement until IRRUT200 completes its processing. IRRUT200 then creates a printout identifying the inconsistencies it found.

   Note: If the RACF data set contains a large number of profiles, the data set might be reserved for a long period of time while the verification is being done.

3. If you specify *only* a work data set on the SYSUT1 DD statement, and do not specify a SYSRACF DD statement, IRRUT200 assumes that a copy of the RACF data set exists in the work data set specified. It is normal to get the informational message, IRR62064, warning you that serialization is not held by the IRRUT200 utility during the verification of the work data set.

   Note: The work data set (SYSUT1) can name an active RACF data set. However, because no serialization is held against the work data set, database updates can be performed against the active RACF data set. IRRUT200 might indicate RACF data set errors that are not really errors. Either repeat the procedure during a time period when no updates will be made to the RACF data set, or use one of the first two methods to verify the RACF data set.

If a RACF data set is RACF-protected, you must have at least Read authority to access the data set. IRRUT200 runs as an APF-authorized program.

When running the IRRUT200 utility under a TMP (terminal-monitor program) that allows multitasking, you cannot have any other active task in your session. Allow IRRUT200 to complete before executing any other TSO command.

IRRUT200 loads a copy of the class descriptor table (CDT) supplied by IBM (ICHRRCDX) and the installation-supplied class descriptor table (ICHRRCDE). If you have not created ICHRRCDE, ignore any system messages (for example, CSV003I) telling you that it has not been found.

Input and output for IRRUT200

IRRUT200 uses the following input:

- A control data set, which contains the utility control statements that indicate the functions to be performed.
- A RACF data set. (This is not required if the work data set already contains a copy.)
- A work data set into which a RACF data set is copied. (This data set is not required if a RACF data set is to be used throughout processing.) Note that this work copy can be used as a RACF data set backup.

IRRUT200 produces the following output:

- A message data set containing diagnostic error messages.
- An output data set for printing statistical data and the results of the IRRUT200 operations.

Control: IRRUT200 is controlled by job control statements and utility control statements. The job control statements are necessary to execute or invoke the program and define the data sets used and produced by the program. The utility control statements (described in “Utility control statements” on page 223) control the functions of the program.

Job control statements: The following job control statements are necessary for using IRRUT200:

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IRRUT200) or, if the job control statements reside in a procedure library, the procedure name.

SYSRACF DD Defines a RACF data set on a direct access device. IRRUT200 requires this data set unless the work data set already contains a copy of the RACF data set.

SYSUT1 DD Defines a work data set on a direct access device. IRRUT200 requires this data set unless a RACF data set is used throughout processing. Do not specify the same data set that you specify for SYSRACF.

Note: Do not specify the RLSE subparameter with the SPACE subparameter on this statement. The RLSE subparameter causes an abend because IRRUT200 uses IEBGENER to copy the RACF data set.

SYSIN DD Defines the control data set. The control data set is normally found in the input stream; however, it can be a member of a procedure library or a sequential data set existing elsewhere.

SYSUT2 DD Defines a sequential message data set.

SYSPRINT DD

Defines a sequential data set for printed output. The data set can be written to an output device, a tape volume, or a direct access device.

Note: We do not recommend running IRRUT200 under TSO, because this can increase the time that the RACF data set is RESERVEd. However if you run IRRUT200 under TSO, you can allocate both the SYSIN and SYSUT2 data sets to the terminal. Although you can allocate SYSPRINT to the terminal, it should be allocated to SYSOUT, because IRRUT200 might produce a large volume of output.

IRRUT200 Example

In the following example, IRRUT200 copies a RACF data set to the SYSUT1 data set. A summary listing of all the index blocks is printed. Any BAM block that contains conflicts is also printed with a table of the locations of the conflicts.

```
//VERIFY    JOB
//STEP      EXEC   PGM=IRRUT200
//SYSRACF   DD     DSN=SYS1.RACF,DISP=SHR
//SYSUT1    DD     UNIT=SYSDA,SPACE=(CYL,(10)),
//                 DCB=(LRECL=4096,RECFM=F)
//SYSUT2    DD     SYSOUT=A
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
    INDEX
    MAP
    END
/*
```

Utility control statements

IRRUT200 is controlled by utility control statements that have the following format. Enter each statement on a separate line. If you enter two statements on the same line, the system ignores the second statement.

Utility control statement for IRRUT200

```
INDEX [FORMAT]
I
```

where:

INDEX specifies you want the index scan function performed.

FORMAT specifies a formatted listing of all the index blocks.

Only one blank can separate INDEX and FORMAT.

You can use only columns 1 through 72.

Utility control statement for IRRUT200

```
MAP [ALL]
M
```

where:

MAP specifies you want the BAM/allocation verification performed.

ALL specifies that you want the encoded map for each BAM block in the RACF data set printed.

Only one blank can separate MAP and ALL.

You can use only columns 1 through 72.

Utility control statement for IRRUT200

```
END
```

where:

END terminates the utility program.

You can use only columns 1 through 72.

Scanning the index blocks

When an index block scan is requested, IRRUT200 verifies that the following are all true:

- The pointer to every index block is a multiple of 4096.
- Every index block begins with the value X'8A'.
- Every index entry name has a valid length.
- Every pointer entry in the index block is preceded by the value X'6'*x* (*x* can be any value).
- Only level-one blocks appear in the sequence set.
- Offsets to the last index entry in each block are correct.
- Offsets to free space in each index block are correct.
- The offset table points to index entries.
- Every regular index entry must have a nonzero segment count.
- Every alias index entry must have a length consistent with the base profile data.

Unformatted printout

If an index block does not meet all the requirements during the verification process, IRRUT200 prints a dump of the block, in hexadecimal. An error message precedes the dump.

Some of these messages are also displayed at your terminal. For an explanation of these messages, see *z/OS Security Server RACF Messages and Codes*.

IRRUT200 provides the following information for each block that is not dumped:

- Title lines identifying the level and relative byte address (RBA) of the index block
- Validity check messages pertaining to the block
- The total number of entry names in the block
- The number of unused bytes in the block
- The average name length in the block
- The level of the block as defined in the header
- The offset to the last entry name in the block
- The offset to free space as defined in the header of the block

Following this information are summary statistics about the index. These statistics might not be representative of the entire RACF data set because they represent only the processed blocks that were not dumped due to errors. The summary statistics are:

- The total number of index entry names in the RACF data set. A name is counted each time it appears in the index.
- The average number of names in each index block
- The average name length in the entire index
- The average number of unused bytes in each index block
- The total number of index blocks
- The total number of level-one index blocks

Formatted printout

If a formatted index scan is requested, IRRUT200 also provides, in addition to the output previously described, a formatted printout of each index block that is not dumped because of an error. The formatted block immediately follows any validity-check messages for that block. The following information is provided for each index entry within a regular index block:

- The offset of the entry within the block
- The front-end compression count
- Index entry name. If the name is followed by a G in parentheses, it is generic. For level-one index blocks, the entry name refers to a profile name (entity name). For upper-level index blocks (not level-one), the entry name corresponds to an entry name in the next-lower level index block, but its suffix might have been truncated or rounded to save space in the database and so the names might not match.
- The RBA of the next-level index block or, for level-one blocks, the RBA of the profile
- The block, byte, and bit of the BAM that describes the storage of the segment pointed to by the RBA.

Note: See Figure 31 on page 226 for sample output that IRRUT200 produces when you request formatted index blocks.

```
                                        ****  INDEX BLOCK VERIFICATION  ****
                                      ****  SCAN OF INDEX BLOCKS AT LEVEL 01  ****

BLOCK WITH RBA OF 00000000E000

OFFSET  COMP.                          ENTRY NAME                                  RBA          BAM
        COUNT                                                                             BLOCK  BYTE BIT
 00E    000  ADAM                                                              00000000DB00   00   02F   3
 026    000  ADRIAN                                                            00000000DC00   00   02F   4
 040    000  ALAN                                                              00000000DD00   00   02F   5
 058    000  ALEX                                                              00000000DE00   00   02F   6
 070    000  ALICIA                                                            00000000DF00   00   02F   7
 08A    000  ANN                                                               00000000F000   00   032   0
 0A1    000  BELINDA                                                           00000000F100   00   032   1
 0BC    000  BETTY                                                             00000000F200   00   032   2
 0D5    000  BILL                                                              00000000F300   00   032   3
 0ED    000  BOB                                                               00000000F400   00   032   4
 104    000  BOB.* (G)                                                         00000000F800   00   033   0
 11E    000  BRAD                                                              00000000F500   00   032   5
 136    000  BRENDA                                                            00000000D400   00   02E   4
 150    000  BRENDA.* (G)                                                      00000000F900   00   033   1
 16D    000  DASDVOL -D001                                                     00000000D000   00   02E   0
 18E    000  DASDVOL -D002                                                     00000000D900   00   02F   1
 1AF    000  DASDVOL -D003                                                     00000000DA00   00   02F   2
 1D0    000  IBMUSER                                                           00000000D700   00   02E   7
 1EB    000  SECLABEL-SYSHIGH                                                  00000000D100   00   02E   1
 20F    000  SECLABEL-SYSLOW                                                   00000000D200   00   02E   2
 232    000  SECLABEL-SYSNONE                                                  00000000D300   00   02E   3
 256    000  SYSCTLG                                                           00000000D600   00   02E   6
 271    000  SYS1                                                              00000000F600   00   032   6
 289    000  VSAMDSET                                                          00000000D500   00   02E   5
 2A5    000  255 X'FF'S
 3B1         SEQUENCE SET POINTER                                             000000000000

TOTAL NAMES IN THIS BLOCK-025. UNUSED BYTES-3092. AVERAGE NAME LENGTH-017.
LEVEL NUMBER-01. DISPLACEMENT TO LAST KEY-03B1. DISPLACEMENT TO FREE SPACE-03BA
(G) - ENTITY NAME IS GENERIC

                                             ****  SEQUENCE SET RBAS  ****

      RBA  00000000E000
                                          ****  INDEX FUNCTION STATISTICS  ****

TOTAL NUMBER OF NAMES IN RACF DATA SET 00000025
TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000001
AVERAGE NUMBER OF NAMES PER INDEX BLOCK 025
AVERAGE NAME LENGTH 017
AVERAGE NUMBER OF UNUSED BYTES PER INDEX BLOCK 3092
TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000001
```

Figure 31. Sample output of formatted index produced by IRRUT200

The following information is provided for each alias index entry within the alias index sequence block set:

- The offset of the entry within the block
- The front-end compression count
- Index entry name. The first 3 bytes of the entry are non-EBCDIC and indicate the characteristics of the index entry. When more than one base profile name is associated with the entry, each base profile name appears on a separate line under the BASE PROFILES column, leaving the other columns blank.
- Count of base profiles associated with the entry
- Base profiles associated with the entry

Note: See Figure 32 for sample output that IRRUT200 produces when you request formatted alias index blocks.

```
                                ****  INDEX BLOCK VERIFICATION  ****
                            ****  SCAN OF ALIAS INDEX BLOCKS AT LEVEL 01  ****

BLOCK WITH RBA OF 000000014000

OFFSET COMP                  ENTRY NAME                                    COUNT OF        BASE PROFILES
        COUNT                                                               BASE PROFS
00E    0000  01030200000001                                               001              GROUP1
02B    0000  01030200000002                                               003              GROUP2
                                                                                          GROUP3
                                                                                          GROUP4
058    0000  01030200000005                                               002              GROUP5
                                                                                          GROUP6
07D    0000  02080200000000                                               001              V10U2028
09C    0000  02080200000001                                               001              UIDUSER1
0BB    0000  02080200000002                                               003              UIDUSER2
                                                                                          UIDUSER3
                                                                                          UIDUSER4
0EE    0000  02080200000005                                               002              UIDUSER5
                                                                                          UIDUSER6
117    0000  020C02alphabetics ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghij     001              LNUSER2
             klmnopqrstuvwxyz
171    0000  020C02ampersand &                                           001              LNUSER4
196    0000  020C02dash -                                               001              LNUSER5
1B6    0000  020C02numbers 0123456789                                   001              LNUSER3
1E2    0000  020C02period .                                             001              LNUSER6
204    0000  020C02underscore _                                         001              LNUSER7
22A    0000  020C02A                                                    001              LNUSER8
245    0000  020C02DB for V10U2028                                      001              V10U2028
26F    0000  020C02THIS IS MY SNAME                                     001              LNUSER1
299    0000  020D02alphabetics ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghij     001              NDSUSER2
             klmnopqrstuvwxyz
2F4    0000  020D02manana ~ exclamation ! atsign @ pound # dollar $     001              NDSUSER4
              percent % not ^ ampersand & left paren ( right paren
             ) underscore _ dash - left set { right set } backslash
              \ less than < greater than > question mark ? period .
              FILLER FILLER       FILL TO THE MAX
405    0000  020D02numbers 0123456789                                   001              NDSUSER3
432    0000  020D02A                                                    001              NDSUSER5
44E    0000  020D02DB for V10U2028                                      001              V10U2028
478    0000  020D02THIS IS MY UNAME                                     001              NDSUSER1
4A3    0000  255 X'FF's
5B0           SEQUENCE SET POINTER

TOTAL NAMES IN THIS BLOCK-023. UNUSED BYTES-2585. AVERAGE NAME LENGTH-037.
LEVEL NUMBER-01. DISPLACEMENT TO LAST KEY-05B0. DISPLACEMENT TO FREE SPACE-05B9

                                     ****  ALIAS SEQUENCE SET RBAS  ****
      RBA  000000014000
                                     ****  INDEX FUNCTION STATISTICS  ****
TOTAL NUMBER OF NAMES IN RACF DATA SET 00000095
TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000004
AVERAGE NUMBER OF NAMES PER INDEX BLOCK 023
AVERAGE NAME LENGTH 039
AVERAGE NUMBER OF UNUSED BYTES PER INDEX BLOCK 2509
TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000001
```

Figure 32. Sample output of formatted alias index produced by IRRUT200

BAM/allocation comparison

When a BAM/allocation comparison is requested, IRRUT200 performs the following verifications:

- Every index entry name must have a valid length.
- If MAP ALL is requested, the names and segment types are checked between the Level 1 regular index and the segments pointed to by the RBAs.
- The logical length of profiles must be a multiple of 256 and must be less than or equal to the allocated length as defined in the header of the profile.
- The actual number of templates must be less than or equal to the space allocated for templates in the inventory control block (ICB).
- The RBA of each template defined in the ICB must have these characteristics:
  - It is a multiple of 4096.
  - The first two bytes are zero.
  - The last four bytes are nonzero.
- The RBA of each BAM block is a multiple of 4096, and its first two bytes are zero.
- The count of BAM blocks in the ICB is greater than zero.
- The number of blocks defined by a BAM block is between 1 to 2008, inclusive.
- Every regular index entry must have a nonzero segment count.
- Every alias index entry must have a length consistent with the base profile data.

When a block does not meet all of these requirements, IRRUT200 prints a dump of the block in hexadecimal. An error message precedes the dump.

Some of these messages are also printed. For an explanation of these messages, see *z/OS Security Server RACF Messages and Codes*.

IRRUT200 produces an encoded map of each BAM block. Each map is identified by a block number and its relative byte address (RBA), and contains byte offsets to the coded masks within the block. The codes indicate the type of block and the types of consistencies or inconsistencies that exist between the actual allocation of data set segments and the status of the segments as defined by the masks in the BAM blocks. Codes that indicate normal conditions and their meanings are as follows:

Symbol	Meaning
*	The segment is defined as allocated by the BAM and is actually allocated.
0	The segment is defined as unallocated by the BAM and is actually unallocated.
B	Refers to a BAM block. This symbol implies an asterisk (*).
F	Refers to the first block (ICB). This symbol implies an asterisk (*).
I	Refers to an index block with the level in the next positions. This symbol implies an asterisk (*).
S	Refers to a segment table block. This symbol implies an asterisk (*).
T	Refers to a template block. This symbol implies an asterisk (*).

/	Undefined space. The BAM block is capable of mapping more space than is defined to the data set. This space is not defined to the RACF data set.

Codes that indicate problems and their meanings are as follows:

Symbol	Meaning
$	Refers to a template or other special block that is defined as unallocated but is actually allocated.
?	Refers to a block that is defined as allocated and is actually allocated. The block is not valid, so its type is unknown.
%	Refers to a block that is defined as unallocated but is actually allocated. The block is not valid, so its type is unknown.
@	The segment is defined as allocated but is pointed to by more than one entry in the index block.
#	The segment is defined as unallocated but is pointed to by more than one entry in the index block.
.	The segment is defined as allocated by the BAM but is actually unallocated. This condition will be corrected the next time you rebuild the data set with the IRRUT400 utility.
+	The segment is defined as unallocated by the BAM but is actually allocated.
–	Refers to an index, BAM, or first block that is defined as unallocated but is actually allocated.

For some of the problem indicator symbols, it might be useful to run the IRRUT400 utility to rebuild the RACF data set. See “Diagnostic capability” on page 236. For other problem indicator symbols, it is necessary to delete the data (using RACF commands or BLKUPD) and then add the data back using RACF commands. For more information about diagnosis, the format of the RACF database, and BLKUPD, see *z/OS Security Server RACF Diagnosis Guide*.

You can use the indicator symbols for normal conditions to determine when it is appropriate to run the IRRUT400 utility to rebuild a RACF data set. You should rebuild a RACF data set when it is running out of usable space, which can occur when there is little space available, or when the space that is available is too fragmented to be usable. The encoded map that IRRUT200 produces with MAP ALL specified specifies what percentage of the data set’s space is in use, indicating how much space is available. You can determine how fragmented the available space is by looking at the map of the BAM blocks. In the example shown in Figure 33 on page 231, the large number of contiguous 0s indicates that there is plenty of contiguous space available in this data set, and little fragmentation. On the other hand, fragmentation would be evident if the mappings that appear as:

```
00000000   00000000   00000000  00000000  ...
```

instead appeared something like:

```
***0****   *0*0****   ***0***0  *00**0**  ...
```

In this fragmented case, profile creations or updates might soon fail because there is not enough contiguous space to accommodate new or larger profiles. Whenever there is little usable space based on the percentage used or based on fragmentation, you should enlarge or rebuild the data set using IRRUT400.

Following the encoded blocks, IRRUT200 prints a table of conflict messages that lists the first 200 locations of possible conflicts in the BAM blocks. These messages locate the inconsistencies by referencing the corresponding block, byte, and bit of the encoded mappings. Each word of the encoded map represents one byte of the BAM block. The relative byte address (RBA) of the storage represented by the bit is also included.

IRRUT200 also provides the following summary statistics concerning the RACF data set:

- The number of BAM blocks defined in the ICB
- The RBA of the last BAM block that defines used space
- The total number of index blocks in the data set
- The total number of level one index blocks
- The number of profiles of each type in the data set
- The percentage of space used in the RACF data set

IRRUT200 produces an encoded map for every BAM block, whether inconsistencies are found or not. As an option, you can request that the encoded maps for an entire RACF data set be printed. If inconsistencies are found, a table of conflict messages follows.

See Figure 33 on page 231 for a sample printout of the encoded map that IRRUT200 produces with MAP ALL specified.

```
                                        ****  BAM BLOCK VERIFICATION  ****
            ****  SYMBOL LEGEND  ****

          *   BAM=ALLOC   , ACTUAL=ALLOC
          0   BAM=UNALLOC , ACTUAL=UNALLOC
          .   BAM=ALLOC   , ACTUAL=UNALLOC
          +   BAM=UNALLOC , ACTUAL=ALLOC
          I   INDEX BLOCK WITH LEVEL IN NEXT POSITIONS
          B   BAM BLOCK
          T   TEMPLATE BLOCK
          S   SEGMENT TABLE BLOCK
          F   FIRST BLOCK (ICB)
          -   BAM=UNALLOC , ACTUAL=ALLOC  I,B,OR F BLK
          $   BAM=UNALLOC , ACTUAL=ALLOC  SPECIAL BLK
          ?   BAM=ALLOC   , ACTUAL=ALLOC  UNKNOWN BLK
          %   BAM=UNALLOC , ACTUAL=ALLOC  UNKNOWN BLK
          @   BAM=ALLOC   , DUPLICATE ALLOCATION
          #   BAM=UNALLOC , DUPLICATE ALLOCATION
          /   UNDEFINED STORAGE
-BLOCK 000 RBA 00000000C000
 014 FFFFFFFF FFFFFFFF TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT
 021 TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT SSSSSSSS SSSSSSSS BBBBBBBB BBBBBBBB
 02E ******** ******** I1111111 11111111 ******** 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 03B 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 048 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 055 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 062 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 06F 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 07C 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 089 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 096 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0A3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0B0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0BD 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0CA 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0D7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0E4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0F1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0FE 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 10B 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 118 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 125 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 132 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 13F //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 14C //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 159 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 166 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 173 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 .
 .
 .
 F91 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F9E //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FAB //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FB8 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FC5 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FD2 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FDF //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FEC //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FF9 //////// //////// //////// //////// //////// //////// ////////

                                        ****  MAP FUNCTION STATISTICS  ****
0NUMBER OF BAM BLOCKS DEFINED 001
 LAST BAM THAT DEFINES USED SPACE - RBA 00000000C000
 RACF DATA SET IS  11 PERCENT FULL.
 TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000001
 TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000001
 NUMBER OF GROUP    ENTRIES - 0000003
 NUMBER OF USER     ENTRIES - 0000013
 NUMBER OF DASDVOL  ENTRIES - 0000003
 NUMBER OF SECLABEL ENTRIES - 0000003
```

Figure 33. Sample output of encoded BAM map produced by IRRUT200

IRRUT200 return codes

The IRRUT200 program sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.
4	(4)	Warning—a validation error was discovered while processing the RACF data set.
8	(8)	A severe error occurred.
C	(12)	An I/O error occurred. If no RACF messages accompany this error, verify that IRRMIN00 has been run against the input data set specified by the SYSRACF DD statement to ensure that it has been properly formatted.
20	(32)	The RACF product is not enabled. For information on enabling RACF, see “Enabling and disabling RACF” on page 62. See *z/OS MVS Product Management* for information on enabling products. After you make the updates required to enable RACF, you must re-IPL in order for the updates to take effect.

Note: See *z/OS Security Server RACF Messages and Codes* for the IRRUT200 messages.

RACF database split/merge/extend utility program (IRRUT400)

IRRUT400 performs the following functions:

- Copies a RACF database to a larger or smaller database, provided there is enough space for the copy.
- Redistributes data from RACF databases. For example, IRRUT400 can split a single data set in the RACF database into multiple data sets, merge multiple data sets in the RACF database (previously split) into fewer data sets, or rearrange RACF profiles across the same number of input and output RACF data sets. Though the utility allows a maximum of 255 input data sets and 255 output data sets, MVS allows RACF to have up to 90 data sets in the primary database and up to 90 corresponding data sets in the backup database.

  Note: IRRUT400 is not intended to be used for merging or rearranging data sets from different systems; the results of doing so are unpredictable.
- Identifies inconsistencies, such as duplicate profiles appearing in different data sets.
- Physically reorganizes the database by bringing all segments of a given profile together.
- Copies a database to a different device type.

Attention: If you have run the IRRIRA00 utility to convert the RACF database to application identity mapping stage 1 or later, you should not use the IRRUT400 utility from a downlevel system.

You should run IRRUT400 when your system has little RACF activity.

RACF sysplex communication

Attention: Whenever you need to run IRRUT400 against a database that is active on a system that is a member of the RACF data sharing group, always run the utility from a system in the group. Failure to do so can cause the utility to build an incorrect output database.

How IRRUT400 works

IRRUT400 formats and initializes each output data set with an ICB, templates, segment table, BAM blocks, a complete index structure, profiles, and an alias index from the input database. It also provides the following features:

- **Index compression:** IRRUT400 builds the index structure of each output data set from the lowest level upwards. Because no block splitting occurs, the index structure is automatically compressed. You can specify a percentage of free space to be left in each index block.
- **Block alignment:** You can request that RACF attempt to keep any segment that is not larger than one block (4KB) within a block boundary.
- **Index structure correction:** The utility uses only the sequence-set index blocks of the input data sets; it builds the output index-structure from the sequence set. As a result, inconsistencies in higher-level index blocks are corrected and do not prevent the utility from executing correctly.
- **Multiple input data sets:** When running with more than one input data set, IRRUT400 copies the ICB, templates, and segment table from the lowest number INDD*n* data set.

Using IRRUT400 to extend a database

Use the IRRUT400 utility to copy an existing RACF database to a larger database.

By using the Split/Merge/Extend utility for an extend operation, you can:
- Compress the index to reduce the number of index blocks
- Place profile records in collating sequence near the appropriate index blocks

Copying a RACF database

As a general rule, running IRRUT200 is the recommended method to create a database copy, if the output database is the same size and on a device with the same track geometry as the input database. However, if you need to produce a copy of a database of a different size from your original, or on a different device type (for example, 3390 to 3380), you *must* use IRRUT400.

In cases where IRRUT200 has detected errors on upper level blocks only, or an analysis of IRRUT200's BAM block mappings has shown that significant fragmentation has occurred, use IRRUT400 to perform the copy. When IRRUT400 copies a database, it rebuilds it, recreating upper level index blocks and reorganizing profiles so that there is no fragmentation. The profile reorganization makes all the segments of a single profile (for example a user profile's base, TSO, and CICS segments) contiguous.

The reorganization that IRRUT400 performs can improve performance by reducing the number of database reads required to read profiles. As a profile is updated over time, its segments are likely to be written to different physical blocks in the database. You can see this by looking at the output of the IRRUT200 INDEX FORMAT function and noting the RBA of each profile segment. RACF reads the

database one 4K block at a time, so the fewer the number of 4K blocks a profile's segments are spread across, the fewer the number of reads required to access all of them, and the better the performance of RACF functions that require database profile access.

For RACF databases consisting of multiple data sets, one IRRUT400 invocation can process one or more of the data sets.

The target of the copy should *not* be an active RACF database. If you need to refresh an active RACF database, use RVARY to deactivate the database before running IRRUT400. After utility processing completes, use RVARY to activate the database.

You can copy an active RACF database, but if you do, you must either specify LOCKINPUT or guarantee that no updates will occur to the input data sets from any system. There are three ways to copy an active database using this utility.

1. Specify the LOCKINPUT parameter.

   This method is preferred. It creates an accurate output database and guarantees that no information is lost before you are able to use the new copy as your active database. Using the LOCKINPUT parameter stops you from writing information, other than statistical updates, to the input database. If you attempt to write to the database while IRRUT400 is running, RACF generates ABEND483 RC50, or ABEND485 RC50 errors.

   Attempts to write to the database result not only from explicit commands like RALTER, but also from a specific logon attempt. For instance, a logon causes a write to the database and fails if:

   - This is your first logon of the day and RACF is not in data sharing mode
   - The password is being changed
   - You are entering the correct password after previously entering an incorrect password

   If the LOCKINPUT keyword is specified, you will be unable to update the input data sets after the execution of this utility. (See "Specifying parameters" on page 238.)

   LOCKINPUT leaves the input database locked to prevent any updates to the input database. If the input database were unlocked when IRRUT400 completed, it might get updated and, therefore, be out of sync with the new copy. If you do not want to switch to the new copy, you must invoke IRRUT400 again with, this time with the UNLOCKINPUT parameter, to unlock the input database so it can be updated.
2. Specify the NOLOCKINPUT parameter.

   Specifying the NOLOCKINPUT parameter does not prevent you from updating the input database.

   - If updates occur to the input database during the copy operation, the results of the utility and the content of the output database are unpredictable. The updates might be successful, an abend might occur, or the output database might become corrupted.
   - If updates occur to the input database after the copy completes, the output database is complete and consistent. However, it does not reflect any of the updates you made to the input database. If you plan to use the output

database and want to avoid losing information, you should be sure that no changes are made between the time that you make the copy and the time RACF begins using it.

3. Use IRRUT200 first, then use IRRUT400, in a two stage process:
   - Stage 1: Use IRRUT200

     Use IRRUT200 to make a copy of a data set from the input database. This copy must be the same size and on a device with the same geometry as the input data set. You can use IRRUT200 only to copy one data set at a time. If the RACF database is comprised of three data sets, for example, you must invoke the utility three times to copy all the data sets.

     Because IRRUT200 uses ENQ or RESERVE serialization while it copies a data set, updates to the data set are delayed briefly until the copy is completed. See “RACF database verification utility program (IRRUT200)” on page 218 for more information.
   - Stage 2: Use IRRUT400

     Use IRRUT400 against the new copy of the database. You can specify the NOLOCKINPUT parameter because the copy is not an active RACF database.

   This option avoids the errors that are possible by using option 1 and avoids the unpredictable results that might occur by using option 2. However, to avoid losing information, you must be sure that no changes are made between the time that you make the copy and the time RACF begins using it.

   If you have a split database, you should not issue any user or group administration commands until all the IRRUT200 copies are complete. Issuing these commands can cause inconsistencies between user and group profiles on the IRRUT400 output database.

Repairing a RACF database

You can use the IRRUT400 utility to repair or reorganize a database that has errors in its upper level index blocks or has fragmented data. In most cases IRRUT200 is the recommended method for copying a database if the database copy is the same size and on a device with the same track geometry. However, when IRRUT200 detects errors on upper level blocks only or an analysis of the IRRUT200 BAM block mappings shows that significant fragmentation has occurred, use IRRUT400 to perform the copy.

In all circumstances you should use IRRUT400 to repair and reorganize the database. When you use IRRUT400 to copy a database, the utility rebuilds the database, recreating upper level index blocks and reorganizing profiles to prevent fragmentation. The profile reorganization makes all the segments of a single profile (for example, a user profile’s base, TSO, and CICS segments) contiguous.

Reorganizing the database can improve performance by reducing the number of database reads required to read profiles. As a profile continues to be updated, its segments are likely to be written to different physical blocks in the database. You can see this in the output in Figure 31 on page 226, noting the RBA of each profile segment. RACF reads the database one 4K block at a time. If the segments are spread across fewer 4k blocks, the number of reads required to access them is less, providing better performance of RACF functions that require database profile access.

Diagnostic capability

IRRUT400 is not designed to provide RACF database diagnostic information. In fact, it is very dependent on the correctness of the RACF database and might abend if corrupted data is encountered. If you suspect a RACF database error, you should start your problem determination by running the IRRUT200 utility and requesting the INDEX and MAP ALL functions. (See “RACF database verification utility program (IRRUT200)” on page 218).

See Chapter 9, “Recovery Procedures,” on page 303 and *z/OS Security Server RACF Diagnosis Guide* for more information on diagnosing and correcting the RACF database.

Additional diagnostic information:

1. IRRUT400 does provide limited diagnosis when using multiple input data sets. In this case, it reports on inconsistencies such as duplicate profiles appearing in different data sets, or defective tape volume sets.
2. In limited situations, IRRUT400 can be used to correct RACF database errors by making a copy of the database. The copy does not contain the same error that the input RACF database contained. This use of IRRUT400 works as long as your database has a valid level-1 (sequence set) structure, and all profile data is valid.

   Therefore, if IRRUT200 reports errors on upper level blocks only—that is, if all profile blocks and level-1 (sequence set) blocks are okay—then IRRUT400 can be used to create a new copy of your RACF database. This works because IRRUT400 does not use the upper-level index blocks. In fact, it reads only the sequence set blocks from the input database and builds new upper level-blocks on the output database. Therefore, your upper-level index block problems can be eliminated by using IRRUT400 to create a new RACF database.

Executing IRRUT400

The following job control statements are necessary for executing IRRUT400:

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IRRUT400) or, if the job control statements are in a procedure library, the procedure name. You can also request IRRUT400 processing options by specifying parameters in the PARM field. See “Specifying parameters” on page 238.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to an output device, a tape volume, or a direct access volume.
INDD*n* DD	Defines a RACF input database. See “Specifying the input database” on page 237.
OUTDD*n* DD	Defines a RACF output database. This statement is not required if you are executing the utility only to identify inconsistencies in a RACF database, or to unlock a database. See “Specifying the output database” on page 237.

If you are redistributing the profiles across more than one data set, you must also provide a *range table.* The range table indicates which profiles are placed in each output data set (using the TABLE keyword to make the determination). See “The

database range table" on page 45 and "Selecting the output data set." If any of the input data sets are RACF-protected, you must have at least UPDATE authority for those data sets.

Note: If the range table is put into a STEPLIB, that STEPLIB must be APF-authorized. If it is not, the STEPLIB is not searched for the range table.

Specifying the input database

Allowable ddnames for the data sets corresponding to the input database are INDD1 through INDD255. The input data sets must be numbered consecutively. For example, if 25 input data sets are provided, they must be assigned ddnames INDD1 through INDD25. The utility processes the input data sets until a number is omitted. You must provide at least one input data set (INDD1).

The only considerations in ordering the input data sets (that is, which data set you assign to INDD1, which to INDD2, and so on) are the following:

- The utility copies the ICB, templates, and segment table from the input data set defined by INDD1 to all output data sets.

  The ICB of the master primary data set contains the setting of the current RACF options. If you want these options copied, INDD1 should be the master primary data set.
- If you do not allow duplicate names in the DATASET class or if duplicate entries exist within any other class, the utility copies the entry from the input data set identified by the lowest-numbered ddname.
- Ensure that each input data set is correctly formatted for the RACF database.
- The IRRUT400 utility does not copy empty data sets; the index must contain valid profile entries for the utility to copy the data set.

Specifying the output database

When redistributing or copying a RACF database, you must code an OUTDD*n* statement for every output data set that the utility will create. The ddnames of the statements defining the output data sets have a relationship to the range table. If a range table is provided, IRRUT400 uses the greatest data set number in the table as an upper boundary for processing. If no range table is provided, the upper boundary is one. IRRUT400 does not process any output data set identified by a ddname with a number greater than the upper boundary. You can specify as many as 255 output data sets on statements named OUTDD1 through OUTDD255.

Output data sets can be new or old direct-access data sets. RACF uses only the first extent. Therefore, do not create any data sets with a secondary space allocation. The output data set cannot be the same as any data set that is pointed to by an INDD statement.

If you are executing IRRUT400 only to identify inconsistencies in the RACF databases, do not code an OUTDD*n* statement. See "Processing of conflicts and inconsistencies" on page 240 for a description of inconsistencies found in the databases.

If you are running IRRUT400 to unlock a database, you do not need an OUTDD*n* statement. See "Specifying parameters" on page 238 for information on the UNLOCKINPUT keyword.

Selecting the output data set

If multiple output data sets are created, the utility uses the range table to determine which profiles to copy to which output data sets. Therefore, you must supply a

range table that indicates the range of profiles to be placed on each data set. You specify the name of the module that contains the range table in the TABLE parameter of the PARM field in the EXEC statement. See “Specifying parameters.” “The database range table” on page 45 describes the format of the range table. If you are using a new range table, you should check that the data set name table (ICHRDSNT) is consistent with the new range table. See “The data set name table” on page 37 for a description of the data set name table.

Processing the output data sets

IRRUT400 initializes each provided output data set as a completely independent RACF database data set, with an ICB, segment table, templates, BAM blocks to describe free space, and an index structure to describe the data set’s contents.

IRRUT400 initialization processing is the same as the processing of the IRRMIN00 utility. As does IRRMIN00, IRRUT400 uses only the first extent of the data set. For old databases, the user must ensure that only one extent is currently allocated for the data set, because the utility cannot detect multiple extents for existing data sets.

The two utilities are different, however, because IRRMIN00 builds the templates using control records read from the IRRTEMP2 CSECT; IRRUT400 merely copies templates read from INDD1. Also, IRRMIN00 builds an ICB with default option settings if PARM=NEW is specified; IRRUT400 copies the option settings from the ICB of INDD1, which is similar to using IRRMIN00 with PARM=UPDATE.

IRRUT400 builds the index of each output data set sequentially from the bottom up. You can request that free space be left in each index block by specifying the FREESPACE parameter. (See “Specifying parameters.”) Specifying FREESPACE allows new entries to be added to a data set when the data set is activated, without causing an index block split.

Aside from the free space requested, the utility compresses the index. IRRUT400 also writes profiles to the output data sets in collating sequence. You can request, with the ALIGN parameter, that no segment of a profile span physical blocks if it can fit into a single physical block (4096 bytes). This decreases the amount of I/O required to access segments that occupy multiple 256-byte slots. The option has no effect on segments that occupy only one slot.

Specifying parameters

You can specify a number of parameters in the PARM field of the EXEC statement of the step executing IRRUT400. The syntax for the parameters is similar to that of the TSO command language. They can be separated by one or more blanks. Embedded blanks are not allowed. Any keyword can be abbreviated to the number of initial characters that uniquely identify that keyword. The specification of redundant or contradictory keywords is considered an error.

LOCKINPUT/NOLOCKINPUT/UNLOCKINPUT

You must specify one of these keywords.

RACF sysplex data sharing

If your system is running in read-only mode, you cannot specify LOCKINPUT or UNLOCKINPUT for IRRUT400.

LOCKINPUT does not allow updates to be made to the specified input data sets, even after the utility terminates. Statistics are updated, however.

If the RACF database is *locked*, a user attempts to logon, and RACF must update the user's profile, the logon may be allowed, or it may fail. It fails if:

- This is the user's first logon of the day and RACF is not in data sharing mode.
- The password is being changed.
- The user is entering the correct password after previously entering an incorrect password.

Otherwise, because RACF is only making a statistical update to the profile, the logon is allowed. In addition, if the user is logging on to TSO, you must ensure that you are running TSO/E 2.3, 2.3.1, or 2.4 with APAR OY56802 applied.

LOCKINPUT locks only the input data sets; it does not lock the output data sets.

Attention

When using LOCKINPUT against an active database, do not schedule maintenance spanning midnight. If the RACF database remains locked past midnight when RACF is not in data sharing mode, users will be unable to submit new jobs or log on, unless you disable the gathering of logon statistics by issuing a SETROPTS NOINITSTATS command. All steps that require a locked database must be performed on the same calendar day.

When you are using LOCKINPUT and running IRRUT400, any activity updating the RACF database will fail with either an ABEND483 RC50 or ABEND485 RC50.

NOLOCKINPUT does not change the status of the data sets, nor does it prevent updates to the input data sets. NOLOCKINPUT is intended to be used for completely inactive RACF databases. If you use it for active RACF databases, all systems sharing the database should have nothing running, such as users logging on, which could result in a write to the active database.

If NOLOCKINPUT is specified and updates occur to the input data sets, the results of the utility and the content of the output data sets will be unpredictable.

UNLOCKINPUT can be used to unlock all data sets that were previously locked by LOCKINPUT. This re-enables your input data set and allows it to be updated.

In most cases, you probably do not need to unlock your input data sets. After using IRRUT400 to create one or more new output data sets, you probably want to use the new output data sets, not the old input data sets. The output data sets are not locked by LOCKINPUT. If, for some reason, the utility is unable to create a valid output data set, it unlocks the input data sets for you. You may need to use UNLOCKINPUT if you mistakenly lock the wrong data set, or if you change your mind after locking a data set.

TABLE(table-name)/NOTABLE

This keyword permits the specification of a user-written range table to be used to select an output data set for each profile. Specifying TABLE(table-name)

indicates that the named load module is to be used. NOTABLE is the default; either specifying or defaulting to it forces the selection of OUTDD1 for all profiles.

If you are using the split or merge option, you must provide a range table (ICHRRNG) to indicate on which data set to place the profiles. The information in the range table must correspond with the information in the data set name table (ICHRDSNT). For further details, see Chapter 3, “RACF customization,” on page 37.

FREESPACE(percent)/NOFREESPACE

This keyword allows you to control the amount of free space left in index blocks created for the output data sets. You can specify that from 0 to 50 percent of the space within the index block is to be left free. The sequence set (level one) will contain the specified percentage of free space; level two will contain one seventh of the specified percentage. Index levels higher than two will contain approximately seven percent free space.

NOFREESPACE [equivalent to FREESPACE(0)] is the default.

The amount of free space you specify should depend on the frequency of updates to the RACF database. For normal RACF database activity, a value of 30 is suggested. If frequent database updates occur, use more.

Note that this keyword does not determine the amount of free space in the database; it affects only the index blocks.

ALIGN/NOALIGN

This keyword allows you to control profile space allocation. Specifying ALIGN forces segments that occupy multiple 256-byte slots to be placed so that they do not span 4096-byte physical blocks. Having a single physical block can decrease the I/O needed to process these segments. Specifying NOALIGN (the default) causes no special alignment.

DUPDATASETS/NODUPDATASETS

This keyword allows you to control the processing of DATASET entries with identical names from different input data sets. Specifying DUPDATASETS indicates that duplicates are allowed and that all DATASET entries are to be processed. If you specify NODUPDATASETS and the utility encounters duplicate entries on different data sets, the utility copies the DATASET entry from the input data set identified by the lowest-numbered ddname. When NODUPDATASETS is in effect, duplicates occurring on a single input data set are all accepted, assuming that they do not conflict with an entry from another data set earlier in the selection sequence. NODUPDATASETS is the default.

Processing of conflicts and inconsistencies

Do not use IRRUT400 to merge data sets from different systems. If you attempt to do so, and the input data sets contain duplicate user, group, or connect profiles with different contents, the output data set will contain inconsistencies.

If more than one input data set is being processed, you can encounter entries with duplicate names. The way in which entries with duplicate names are processed depends on which classes they belong to. If the utility encounters duplicate names across classes or within classes other than DATASET, it copies the entry from the input data set identified by the lowest-numbered ddname and issues a message indicating the entry that was not copied. See the description of the DUPDATASETS/NODUPDATASETS parameter for information on how duplicate DATASET entries are handled.

The possibility of conflicts between tape volume sets exists even when only one input data set is specified. The utility detects conditions described in the following list:

- If more than one output data set is specified, a tape volume set might contain members that are assigned to different output data sets by the range table. Because of the way that tape volume sets are implemented, it is impossible to reconstruct such a tape volume set on the output data sets. Therefore, IRRUT400 does not copy the entire tape volume set, including all its members, and issues a message to that effect.
- It is possible for two tape volume sets to contain one or more members with the same names. Usually, this happens only when more than one input data set is specified. (Because of an internal inconsistency however, it is possible for it to happen even with only one input data set.) It is impossible for IRRUT400 to copy both of these tape volume sets to the output data set. A message is issued to indicate which data set is not copied.

If a logical error exists in an input data set (for example, an index entry not pointing to the correct profile for the entity), IRRUT400 issues an error message because it is impossible to copy the entity to an output data set.

IRRUT400 return codes

The codes returned to the caller by the split/merge/extend utility are:

Hex	(Decimal)	Meaning
0	(0)	Successful completion without error.
4	(4)	A warning condition occurred for one of the following reasons: • Duplicate names that IBM defined caused one or more warning conditions. • An expected output DD statement (OUTDD*n*) was not found. IRRUT400 continued processing. Note that if you intentionally did not specify an output DD statement, IRRUT400 continues processing to identify inconsistencies in the RACF database, and you can ignore this warning.
8	(8)	One or more error conditions occurred because of one of the following conditions: • Duplicate names that IBM did not define • A defective tape-volume set
C	(12)	One or more severe error conditions resulted from an error on an output data set.

Hex	(Decimal)	Meaning
10	(16)	A terminating error condition occurred for one of the following reasons: • A recovery environment could not be established. • The SYSPRINT file could not be opened. • An error was found in a parameter specification. • A range table was requested but could not be loaded. • An error was detected in the specified range table. • An error occurred on an input data set. • LOCKINPUT or UNLOCKINPUT was specified when the system was running in read-only mode. • An error related to the coupling facility occurred during LOCKINPUT or UNLOCKINPUT processing. • All input data sets contained valid ICBs, but none contained profiles.
20	(32)	The RACF product is not enabled. As a result, the utility was not run and the RACF database was not formatted. For information on enabling RACF, see “Enabling and disabling RACF” on page 62. See *z/OS MVS Product Management* for information on enabling products. After you make the updates required to enable RACF, you must re-IPL in order for the updates to take effect.

IRRUT400 examples

These examples show how to code the split/merge/extend utility (IRRUT400) to perform different functions.

The examples show the recommended setting for DSORG, DCB=DSORG=PSU. By making the data sets “unmovable,” the installation can assure that utilities do not move the RACF database from the location that RACF placed in its control blocks at the time RACF last opened the database. For example, DFDSS might be run to reclaim (DEFRAG) fragmented space on a volume.

Results are unpredictable if the database is moved from where RACF thinks it is. One possibility is failure of all requests for RACF services, because logical I/O errors will be reported by RACF. You may also lose updates to profiles during an IPL.

If an installation can put in place procedural controls that guarantee that the RACF database will not be moved unless an RVARY INACTIVE command is issued, the installation may choose to make the RACF data sets movable. The installation assumes the risk if the procedural controls fail.

Example 1. Copying a database

In this example, IRRUT400 copies the profiles from a single input data set to a single output data set. This one-to-one copy does not require a range table, because all profiles are directed to OUTDD1. IRRUT400 compresses the output data set, which might be larger or smaller than the input data set, and might even reside on a different type of device. The index blocks of the output data set will contain free space to allow for expansion.

```
//J1       JOB
//         EXEC PGM=IRRUT400,PARM='NOLOCKINPUT,FREESPACE(20)'
//SYSPRINT  DD  SYSOUT=A
//INDD1     DD  DSN=SYS1.RACF5,DISP=OLD
//OUTDD1    DD  DSN=SYS2.RACF5,DISP=(,KEEP),
//              VOL=SER=VOL1,
//              SPACE=(CYL,10,,CONTIG),
//              DCB=DSORG=PSU,
//              UNIT=SYSDA
```

Example 2. Splitting a database

In this example, IRRUT400 splits a RACF database containing a single data set into three output data sets. IRRUT400 assigns profiles to the output data sets using a range table in a module named SELECT. The load module resides in library INSTALL.LINKLIB.

```
//J2      JOB
//        EXEC  PGM=IRRUT400,PARM='NOLOCKINPUT,TABLE(SELECT)'
//SYSPRINT  DD  SYSOUT=A
//INDD1     DD  DSN=SYS1.RACF,DISP=OLD
//OUTDD1    DD  DSN=SYS2.RACF1,DISP=(,KEEP),
//              UNIT=SYSDA,VOL=SER=VOL1,
//              DCB=DSORG=PSU,
//              SPACE=(CYL,5,,CONTIG)
//OUTDD2    DD  DSN=SYS2.RACF2,DISP=(,KEEP),
//              UNIT=SYSDA,VOL=SER=VOL2,
//              DCB=DSORG=PSU,
//              SPACE=(CYL,20,,CONTIG)
//OUTDD3    DD  DSN=SYS2.RACF3,DISP=(,KEEP),
//              UNIT=SYSDA,VOL=SER=VOL3,
//              DCB=DSORG=PSU,
//              SPACE=(CYL,5,,CONTIG)
//STEPLIB   DD  DSN=INSTALL.LINKLIB,DISP=SHR
```

Example 3. Merging data sets

In this example, IRRUT400 merges two previously-split data sets from the same system into a single data set. IRRUT400 first makes a test run to identify any possible inconsistencies. Data set entries with identical names, but from different RACF data sets, are allowed.

```
//J3A       JOB
//          EXEC   PGM=IRRUT400,PARM='NOLOCKINPUT,DUPDATASETS'
//SYSPRINT    DD   SYSOUT=A
//INDD1       DD   DSN=SYS1.RACF1,DISP=OLD
//INDD2       DD   DSN=SYS1.RACF2,DISP=OLD
```

After any identified inconsistencies are corrected, IRRUT400 performs the actual merge. To improve I/O performance, the utility is to align profiles written to the output data set.

```
//J3B     JOB
//        EXEC   PGM=IRRUT400,PARM=('NOLOCKINPUT,DUPDA FREE(10)',ALIGN)
//SYSPRINT     DD SYSOUT=A
//INDD1        DD DSN=SYS1.RACF1,DISP=OLD
//INDD2        DD DSN=SYS1.RACF2,DISP=OLD
//OUTDD1       DD DSN=SYS2.RACF,DISP=(,KEEP),
//             UNIT=SYSDA,VOL=SER=VOL1,
//             DCB=DSORG=PSU,
//             SPACE=(CYL,10,,CONTIG)
```

Example 4. Copying to a larger database

The active RACF database is full, and requests are failing because of lack of space. In this example, IRRUT400 copies the RACF database to a larger data set, rebalances the index structure, and provides room in each index block for expansion. IRRUT400 is to align the profiles to improve access time for some of the

larger profiles. Because LOCK is specified for PARM on the EXEC statement in this example, no new entries can be made to the input database unless you use the UNLOCK PARM on IRRUT400 to reset the lock bit in the inventory control block (ICB) and unlock the database. However, because the input database is full, you probably don't want to make new entries to it. You want to make new entries to the new, larger, output database, which isn't locked.

```
//J4      JOB
//        EXEC   PGM=IRRUT400,PARM='LOCK,F(10),A'
//SYSPRINT    DD SYSOUT=A
//INDD1       DD DSN=SYS1.RACF,DISP=OLD
//OUTDD1      DD DSN=SYS2.RACF,DISP=(,KEEP),
//               UNIT=SYSDA,VOL=SER=VOL1,
//               DCB=DSORG=PSU,
//               SPACE=(CYL,15,,CONTIG)
```

Example 5. Unlocking a database

In this example, you intended to run IRRUT400 to lock your test database, but you accidentally locked your production database instead. You want to unlock the database as quickly as possible. IRRUT400 unlocks the database so it can resume normal RACF database update processing. Because the database is not being copied, IRRUT400 can unlock it quickly.

```
//J5      JOB
//        EXEC   PGM=IRRUT400,PARM='UNLOCKINPUT'
//SYSPRINT    DD SYSOUT=A
//INDD1       DD DSN=SYS1.RACF,DISP=OLD
```

Example 6. Copying using a two—stage option

In this example, you want to copy the active RACF database to a larger database at a time when there is still some activity on your system that might result in updates to the RACF database. Because you are using the copy on a test system, not on a production system, The IRRUT400 copy does not need to be an exact copy of the active RACF database.

You do not want to use LOCKINPUT because you do not want to receive errors that result from update attempts to the active database while LOCKINPUT is in effect. Neither do you want to be subject to the unpredictable results that using NOLOCKINPUT can cause. You prefer to use a two-stage process, first using IRRUT200, then IRRUT400. You understand that the active database that is used as input to IRRUT200 and the output database from IRRUT400 will be out-of-synch if updates occur during that interval and that you will lose those updates if you decide to use the IRRUT400 copy as your active database.

```
//VERIFY    JOB
//STEP1     EXEC   PGM=IRRUT200
//SYSRACF   DD     DSN=SYS1.RACF,DISP=SHR
//SYSUT1    DD     DSN=TEMP.CPY200,DISP=OLD
//SYSUT2    DD     SYSOUT=A
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
    END
/*
//STEP2    EXEC PGM=IRRUT400,PARM='NOLOCKINPUT,FREESPACE(20)'
//SYSPRINT  DD  SYSOUT=A
//INDD1     DD  DSN=TEMP.CPY200,DISP=OLD
//OUTDD1    DD  DSN=SYS2.RACF,DISP=(,KEEP),
//              VOL=SER=VOL1,
//              SPACE=(CYL,10,,CONTIG),
//              DCB=DSORG=PSU,
//              UNIT=SYSDA
```

Utilities documented in other documents

The utilities described in this section can be found in other documentation.

RACF database unload utility program (IRRDBU00)

The RACF database unload utility unloads the RACF database to a sequential file. For information on how to use IRRDBU00, see *z/OS Security Server RACF Macros and Interfaces* and *z/OS Security Server RACF Security Administrator's Guide*.

RACF remove ID utility (IRRRID00)

The RACF remove ID utility (IRRRID00) processes the output of the RACF database unload utility (IRRDBU00) and creates commands to remove references in the RACF database to user IDs and group names that are no longer in the database. Alternatively, it can create commands to delete references in the RACF database to specified user IDs and group names. For information on how to use the RACF remove ID utility, see *z/OS Security Server RACF Security Administrator's Guide*.

RACF SMF data unload utility program (IRRADU00)

The RACF SMF data unload utility enables installations to create a sequential file from the SMF security-relevant audit data. The sequential file can be used in several ways: viewed directly, used as input for installation-written programs, and manipulated with sort/merge utilities. It can also be uploaded to a database manager (for example, DB2) to process complex inquiries and create installation-tailored reports. For information on how to use the SMF data unload utility, see *z/OS Security Server RACF Auditor's Guide*.

BLKUPD command

The BLKUPD command modifies the records in a RACF database. You execute BLKUPD as a TSO command. You can use BLKUPD to correct inconsistencies that IRRUT200 finds in the RACF database. For information on how to use BLKUPD, see *z/OS Security Server RACF Diagnosis Guide*.

Data security monitor (DSMON)

The Data Security Monitor produces reports on the status of the security environment at your installation and, in particular, on the status of resources that RACF controls. You can use the reports to audit the current status of your installation's system security environment by comparing the actual system characteristics and resource-protection levels with the intended characteristics and levels. You can also control the reporting that DSMON does by specifying control statements that request certain functions for user input. For information on how to use DSMON, see *z/OS Security Server RACF Auditor's Guide*.

RACF report writer (RACFRW)

The RACF report writer lists the contents of System Management Facilities (SMF) records in a format that is easy to read. You can tailor the reports to select specific SMF records that contain certain kinds of RACF information. For information on how to use the RACF report writer, see *z/OS Security Server RACF Auditor's Guide*.

RRSF VSAM file browser (IRRBRW00)

The RRSF VSAM file browser (IRRBRW00) transcribes workspace data set VSAM file records into a browsable output data set. It is provided in case off-line diagnosis of the RRSF workspace data sets is required. See the RACJCL member of SYS1.SAMPLIB for instructions on running the utility, and *z/OS Security Server RACF Diagnosis Guide* for information on setting up proper security to control its use.

RACFICE reporting tool

The RACFICE reporting tool allows an installation to create tailored RACF reports without requiring a relational database management product, and provides an alternative to the RACF report writer. It makes use of the DFSORT™ ICETOOL reporting facility. RACF makes several ICETOOL-based reports available in SYS1.SAMPLIB. The RACJCL member of SYS1.SAMPLIB provides sample JCL to allocate a report data set and add the RACFICE reports in IEBUPDTE format. The RACFICE member provides the IEBUPDTE-format ICETOOL and DFSORT control statements that implement the RACFICE reports.

For more information on using RACFICE, see *z/OS Security Server RACF Auditor's Guide* and *z/OS Security Server RACF Security Administrator's Guide*.

Chapter 8. RACF installation exits

Overview 249
RACF exits report 250
Extended addressing for exits 250
Data set naming convention table 251
Exits running in the RACF subsystem address space 252
Possible uses of RACF exits 253
Summary of installation-exit callers 253
ACEE compression/expansion exits 255
Range tables 256
Range table example 257
IRRACX01 258
Installing the exit routine 258
Exit recovery 258
Exit routine environment 258
Exit routine processing 258
Programming considerations 259
Entry specifications 259
Return specifications 260
Coded example of the exit routine 260
IRRACX02 260
Installing the exit routine 260
Exit recovery 260
Exit routine environment 260
Exit routine processing 260
Programming considerations 260
Entry specifications 261
Return specifications 261
Coded example of the exit routine 261
Command exits for specific commands 261
ICHCNX00 processing 262
Return codes from the command-preprocessing exit ICHCNX00 264
ICHCCX00 processing 265
Return codes from the command-preprocessing exit ICHCCX00 266
Common command exit 266
Controlling the exit routine through the dynamic exits facility 266
Replacing the exit routine 267
Exit routine environment 267
Exit recovery 267
Exit routine processing 267
Information passed in the parameter list 267
The preprocessing call 269
The postprocessing call 269
Programming considerations 270
Entry specifications 270
Registers at entry 270
Parameter descriptions 270
Return specifications 270
Registers at exit 271
Coded example of the exit routine 271
New-password exit 271
ICHPWX01 processing 272
Return codes from the new-password exit 273
Possible use of the exit 274

© Copyright IBM Corp. 1994, 2004

Password quality control 274
Password authentication exits 275
ICHDEX01 276
Installing the exit routine 276
Exit recovery 276
Exit routine environment 276
Exit routine processing 276
Programming considerations 276
Entry specifications 276
Return specifications 277
Coded example of the exit routine 277
ICHDEX11 278
Installing the exit routine 278
Exit recovery 278
Exit routine environment 278
Exit routine processing 278
Programming considerations 278
Entry specifications 278
Return specifications 279
Coded example of the exit routine 279
RACROUTE REQUEST=AUTH exits 279
Extended addressing 279
Preprocessing exit (ICHRCX01) 280
Return codes from the RACROUTE REQUEST=AUTH preprocessing exit 280
Postprocessing exit (ICHRCX02) 281
Return codes from the RACROUTE REQUEST=AUTH postprocessing exit 282
Possible uses of the exits 282
Allowing access when RACF is inactive 282
Protecting the user's resources from the user 282
Controlling access of shared user IDs 283
RACROUTE REQUEST=DEFINE exits 283
Extended addressing 284
Automatic direction of application updates 284
Preprocessing exit (ICHRDX01) 284
Return codes from the RACROUTE REQUEST=DEFINE preprocessing exit 285
Postprocessing exit (ICHRDX02) 285
Return codes from the RACROUTE REQUEST=DEFINE postprocessing exit 285
RACROUTE REQUEST=FASTAUTH exits 286
Preprocessing exits (ICHRFX01 and ICHRFX03) 286
ICHRFX01 286
ICHRFX03 288
Postprocessing exits (ICHRFX02 and ICHRFX04) 290
ICHRFX02 291
ICHRFX04 293
Possible uses of the exits 295
Controlling access of shared user IDs 295
RACROUTE REQUEST=LIST exits 296
Pre- and postprocessing exit (ICHRLX01) 297
Return codes from ICHRLX01 297
Selection exit (ICHRLX02) 297
Return codes from the RACROUTE REQUEST=LIST selection exit 298
RACROUTE REQUEST=VERIFY(X) exits 298
Preprocessing exit (ICHRIX01) 299

Return codes from the RACROUTE REQUEST=VERIFY(X) preprocessing exit 299
Postprocessing exit (ICHRIX02) 300
Return codes from the RACROUTE REQUEST=VERIFY(X) postprocessing exit 301
RACF report-writer exit 301
ICHRSMFE processing 302
Return codes from the RACF report-writer exit (ICHRSMFE) 302
SAF router exits 302

This chapter documents the installation exits and gives associated guidance information. The installation exits are product-sensitive programming interfaces.

Overview

RACF provides a number of installation exits that enable you to use your own routines to enhance the facilities offered by RACF, as well as to optimize its usability. For RACROUTE requests, the exits allow an installation to tailor the parameters passed on the macro and to perform any additional security checks or processing that the installation requires.

The RACF initialization routine loads the exit routines during system IPL and, except for IRREVX01, places the exit addresses in the RACF communication vector table (RCVT). If RACF determines (through a search of the LPA) that the exit routines were not supplied, RACF sets the RCVT fields pointing to the exit routines to zero. If you change an exit, except IRREVX01, you must re-IPL MVS for the changes to take effect. IRREVX01 is defined to the dynamic exits facility, and you can update it without re-IPLing.

RACF initialization message ICH508I displays the names of the exits that are active for the IPL. Because IRREVX01 is defined to the MVS dynamic exits facility, if its name appears in the ICH508I message at least one active routine has been added to the IRREVX01 exit point at this particular time in the IPL.

The exit routines must be reenterable and refreshable and must be located in the link-pack area: PLPA, FLPA, or MLPA. The exit routines receive control with standard linkage conventions; the exit routines should use standard linkage conventions to return control.

Register contents upon entry to the RACF exits (except for RACROUTE REQUEST=FASTAUTH requests) are:

R0 Unknown
R1 Address of exit parameter list
R2—R12
Unknown
R13 Address of save area
R14 Return address
R15 Address of exit

RACF uses the first word of the save area pointed to by register 13. Exits must not modify this part of the save area.

When the preprocessing exit routines for RACROUTE requests receive control, RACF has already validity-checked the macro parameters, but has not yet performed any other processing.

Make changes or additions to the parameter information only in the designated areas. In most cases, if a pointer is provided in the parameter list you can modify the data that it is pointing to; if the parameter list contains a 0 pointer, you can supply data, and then change the pointer to address the data.

There are special considerations for exits when automatic direction is active. For information on these considerations, see “Installation exit considerations” on page 149.

See *z/OS Security Server RACF Data Areas* for a mapping of the accessor environment element (ACEE) data area, which is helpful when you code exit routines.

RACF exits report

The data security monitor (DSMON) produces the RACF exits report. This report lists the names of all the installation-defined RACF exit routines and specifies the size of each exit routine module. If the RACF communications vector table (RCVT), which contains the address of each RACF exit-routine module except IRREVX01, indicates that an exit-routine module should exist, but the module cannot be loaded, or that the entry address does not correspond with the address specified in the RCVT, DSMON prints an error message.

DSMON lists IRREVX01 when at least one active exit routine is defined at the time the report is created. The report does not include the routine names or sizes, and lists the length of IRREVX01 as ″NA″ (not available).

Note: You must have the AUDITOR attribute to run the report. See *z/OS Security Server RACF Auditor's Guide* for more information.

You can use the information in this report to verify that only those exit routines that have been defined by your installation are active. The existence of any other exit routines might indicate a system security exposure, because RACF exit routines could be used to bypass RACF security checking.

Similarly, if the length of an installation-defined exit routine module differs from the length of the module when it was defined by your installation, you should notify your RACF security administrator, because the module might have unauthorized modifications.

Extended addressing for exits

RACF supports callers running in either AMODE(24) or AMODE(31), except for RACROUTE REQUEST=SIGNON, which requires its callers to be in AMODE(31). If all of your RACF exits have AMODE(31) or AMODE(ANY), parameters and parameter lists for the exits can reside above the 16MB address. If you have AMODE(24) exits, RACF data areas will be placed below the 16MB address in storage to ensure that your exits process correctly. The RCVT, any task-level ACEEs (pointed to by TCBSENV), and the address space ACEE (pointed to by ASXBSENV) always reside below 16MB.

For best use of virtual storage and best RACF performance, we recommend that you write and link-edit your exits to run in AMODE(31) or AMODE(ANY).

During system IPL, RACF initialization records the AMODE of each installation exit routine it finds. All RACF exit routines, except the RACROUTE

REQUEST=FASTAUTH exits, are called in their defined AMODE and must return control to RACF in the mode in which they were called.

RACF calls the RACROUTE REQUEST=FASTAUTH exits and MVS calls the SAF router exits in the AMODE (24 or 31) used by the invoker of the service. For this reason, the RACROUTE REQUEST=FASTAUTH and SAF router exits must be capable of running in either mode.

In many cases RACF must copy caller-supplied parameter areas to an area below 16MB, so that the AMODE(24) exit can address the parameter areas. However, when the RACROUTE REQUEST=AUTH or RACROUTE REQUEST=DEFINE exit routines receive the ACCLVL parameter or INSTL parameters, RACF does not know the format or length of the parameters being passed, and therefore cannot copy the parameters into 24-bit storage. Because the parameter list pointing to these parameters is in 24-bit storage, you must modify the exit routines to handle the parameters if the exit routines access these areas, and if they are passed by callers in 31-bit mode.

If a 31-bit caller issues a RACROUTE REQUEST=VERIFY request with the ACEE parameter, and does not specify that the ACEE should be placed below 16MB, it is placed above 16MB, unless a RACF exit is marked (or defaulted to) AMODE(24).

Any data areas attached to an ACEE that is below 16MB will also be below 16MB, with two exceptions:

- The list of generic profiles can reside above 16MB if no RACF exits have AMODE(24).
- The list of RACLISTed profiles.

For an ACEE other than an address space ACEE, if a 31-bit caller of RACROUTE REQUEST=LIST requests that the RACLISTed profiles be placed above 16MB, and no installation-supplied RACF exits have AMODE(24), the profiles are placed above 16MB by RACROUTE REQUEST=LIST.

Data set naming convention table

The data set name format used by RACF is based on the TSO data set naming rules (see *z/OS TSO/E User's Guide*). In addition to these rules, RACF requires the data set name to have at least two qualifiers and the high-level qualifier to be a valid RACF-defined user ID or group name.

RACF allows installations to create a naming convention table (ICHNCV00) that RACF uses to check the data set name in all the commands and RACROUTE requests that process data set names. This table helps an installation set up and enforce data set naming conventions that are different from the standard RACF naming conventions.

You create the naming convention table by using the ICHNCONV macro. (For information on coding the ICHNCONV macro, see *z/OS Security Server RACF Macros and Interfaces*.)

If the required processing is too complex to be handled by using the ICHNCONV macro, you can use exit routines to modify data set naming conventions. A naming convention routine or table should be able to perform the following functions:

- Conversion of a user's real data set name into a format acceptable to RACF (with the high-level qualifier as a user ID or a group name)

- Conversion of the internal RACF format back to the user's real data set name for display purposes (through IRRUT100, LISTDSD, and SEARCH after a profile is located but before it is displayed)
- Identification of a user ID or group name to be used for authority checking
- Optionally, enforce other restrictions on data set names (format and content) on define requests (such as ADDSD, RACROUTE REQUEST=DEFINE TYPE=RENAME)

RACF processes the naming convention table before it calls the following exit routines:

- ICHRDX01, RACROUTE REQUEST=DEFINE preprocessing exit routine
- ICHRCX01, RACROUTE REQUEST=AUTH preprocessing exit routine
- ICHCNX00, command naming-convention exit routine
- ICHCCX00, command naming-convention exit routine
- The postprocessing call to IRREVX01, common command exit routine

You can use the exits for additional processing of data set names.

The RACF initialization routine finds the ICHNCV00 module and stores the address in the RCVT. If the initialization routine finds the module, ICHNCV00 is listed in message ICH508I with the other exit routines.

If you change ICHNCV00, you must reassemble it, link-edit it, and re-IPL.

ICHNCV00 has AMODE(31) and RMODE(ANY). You cannot override these values.

See also "Command exits for specific commands" on page 261.

Exits running in the RACF subsystem address space

RACF commands can run in the RACF subsystem address space instead of the user's address space. RACF commands run in the RACF subsystem address space when:

- They are directed using command direction or automatic command direction.
- They are issued as operator commands.

Application updates can also run in the RACF subsystem address space instead of the user's address space when they are automatically directed. Application updates that invoke exits include:

- RACROUTE REQUEST=DEFINE
- RACDEF
- RACROUTE REQUEST=EXTRACT,TYPE=REPLACE
- RACXTRT specifying TYPE=REPLACE

An installation exit that is sensitive to where it is running can check the ACEERASP bit in the ASXB-level ACEE to determine whether it has gained control in the RACF subsystem address space. Some examples of situations where exits need to be sensitive to where they are running are:

- When an exit associated with a command runs in the user's address space, it can issue a message to the user via TPUT or PUTLINE (for a command issued by a TSO user) or via PUTLINE or WTO (for a command issued by a batch TSO job). However, if the command is running in the RACF subsystem address space, only PUTLINE works. For exits ICHCCX00 and ICHCNX00, RACF provides sufficient information to allow an exit to use PUTLINE. For exit IRREVX01, RACF provides a message area where the exit can provide text to be inserted in a

message. Exits other than these should recognize when they are running in the RACF subsystem address space and avoid trying to communicate with the user.

- If an exit wants to find the user's ACEE and it is running in the user's address space, it can generally use the ASXBSENV pointer to find the address-space-level ACEE, and can ignore the TCBSENV pointer (TCB-level ACEE). This is not correct programming, but generally works. However, if the exit is running in the RACF subsystem address space, the exit *must* first look for a TCB-level ACEE if it needs to find the user's ACEE.

Possible uses of RACF exits

Some possible uses of the RACF exits are described with the individual exit. They are:

- "Password quality control" on page 274
- "Allowing access when RACF is inactive" on page 282
- "Protecting the user's resources from the user" on page 282
- "Controlling access of shared user IDs" on page 283
- "Controlling access of shared user IDs" on page 295

Summary of installation-exit callers

Table 12 and Table 13 on page 254 list the macros, commands, and utilities that give control to each exit routine.

Table 12. RACF installation-exits cross-reference table—Part 1 of 2

Functions	ICHNCV00	ICHRDX01 ICHRDX02	ICHRCX01 ICHRCX02	IRRACX01 IRRACX02	ICHRIX01 ICHRIX02
ADDSD	X (Note 1, Note 2)	Note 1	Note 2		
ALTDSD			Note 2		
ALTUSER					
DELDSD	X (Note 1, Note 2)	Note 1	Note 2		
DELGROUP					
DELUSER					
LISTDSD	X (Note 2)		Note 2		
PASSWORD					
PERMIT	X (Note 2)		Note 2		
RALTER			Note 2		
RDEFINE		Note 1	Note 2		
RDEFINE FROM (on data sets)	X (Note 1, Note 2)	Note 1	Note 2		
RDELETE		Note 1	Note 2		
REMOVE					
RLIST			X		
SEARCH	X		X	Note 3	Note 3
SETROPTS RACLIST				Note 3	Note 3
RACROUTE REQUEST= DEFINE	X	X	Note 2	Note 3	Note 3

Table 12. RACF installation-exits cross-reference table—Part 1 of 2 (continued)

Functions	ICHNCV00	ICHRDX01 ICHRDX02	ICHRCX01 ICHRCX02	IRRACX01 IRRACX02	ICHRIX01 ICHRIX02
RACROUTE REQUEST=AUTH	X		X	Note 3	Note 3
RACROUTE REQUEST= VERIFY			Note 2	X	X
RACROUTE REQUEST= LIST					
RACROUTE REQUEST= FASTAUTH					
RACROUTE REQUEST= EXTRACT			Note 2	X	
RACROUTE REQUEST= SIGNON				Note 3	Note 3
IRRUT100	X				

Note:

1. The function invokes RACROUTE REQUEST=DEFINE processing and could be affected by the DEFINE request exits.

2. The function can invoke RACROUTE REQUEST=AUTH processing and could be affected by the AUTH request exits.

3. The function can invoke RACROUTE REQUEST=VERIFY processing and could be affected by the VERIFY request exits.

Table 13. RACF installation-exits cross-reference table—Part 2 of 2

Function	ICHRLX01 ICHRLX02	ICHRFX01 ICHRFX02 ICHRFX03 ICHRFX04	ICHPWX01	ICHCNX00	ICHCCX00	IRREVX01
ADDSD				X		X
ALTDSD				X		X
ALTUSER			X			X
DELDSD				X		X
DELGROUP					X	X
DELUSER					X	X
LISTDSD				X		X
PASSWORD			X			X
PERMIT				X		X
RALTER	Note 4	Note 5				X
RDEFINE	Note 4	Note 5		X		X
RDEFINE FROM (on data sets)						X
RDELETE						X

Table 13. RACF installation-exits cross-reference table—Part 2 of 2 (continued)

Function	ICHRLX01 ICHRLX02	ICHRFX01 ICHRFX02 ICHRFX03 ICHRFX04	ICHPWX01	ICHCNX00	ICHCCX00	IRREVX01
REMOVE					X	X
RLIST						X
SEARCH				X		X
SETROPTS RACLIST	Note 6					X
RACROUTE REQUEST= DEFINE						
RACROUTE REQUEST= AUTH						
RACROUTE REQUEST= VERIFY			X			
RACROUTE REQUEST= LIST	X					
RACROUTE REQUEST= FASTAUTH		X				
RACROUTE REQUEST= EXTRACT				X		
RACROUTE REQUEST= SIGNON						
IRRUT100				X		

Note:

4. When the user is not SPECIAL and has specified ADDMEM and DELMEM, this function invokes RACROUTE REQUEST=LIST processing and could be affected by the LIST request exits.

5. When the user is not SPECIAL and has specified ADDMEM and DELMEM, the function invokes RACROUTE REQUEST=FASTAUTH and could be affected by the FASTAUTH request exits.

6. The function invokes RACROUTE REQUEST=LIST processing and could be affected by the LIST request exits.

ACEE compression/expansion exits

When RACF compresses an ACEE, it stores the ACEE as one contiguous area, called an ENVR object, containing no pointers. ACEEIEP can point to various user-defined data structures which can be non-contiguous. Because the data that needs to be saved is pointed to in nonstandard ways, RACF provides two exits that allow an installation to tell RACF what data to save, in addition to the ACEE itself. The two exits are:

- IRRACX01, for task mode, non-cross-memory environments
- IRRACX02, for cross-memory environments and SRB mode

These exits receive the same parameter list, do the same processing, and return the same output.

These exits are called as part of compressing or expanding an ACEE, which can occur during the processing of some commands and RACROUTE requests (see Table 12 on page 253 for details) and during the processing of the initACEE callable service.

It is expected that most installations will not have to code the ACEE compression/expansion exits. Installations that do not use ACEEIEP, and installations that have ACEEIEP pointing to standard data in RACF's standard format, do not need to provide these exits. However, if an installation is making a nonstandard use of ACEEIEP, in task mode and non-cross-memory environments it can use the ACEE compression/expansion exits to ensure that the compressed or expanded ACEE contains the installation's data. Note, however, that the exits do not get control for ACEE expansion in SRB mode or cross memory mode. In these cases, the installation should have ACEEIEP point only to standard data.

Standard data for ACEEIEP:

Standard data for ACEEIEP has the following characteristics:

1. The first word of the data has the subpool in the first byte, and the length in the last three bytes.

   If you violate this characteristic, you must provide the IRRACX01 and IRRACX02 exits to ensure that RACF processing does not abend when trying to process ACEEIEP during ACEE compression/expansion.
2. The remaining data should be relocatable. This means that the remaining data does not contain any addresses to other data—neither within the block pointed to by ACEEIEP nor in another area of storage.

   If you violate this characteristic, you should provide the IRRACX01 and IRRACX02 exits to ensure that the compressed or expanded ACEE contains all of your data.

Nonstandard use of ACEEIEP:

Examples of nonstandard use of ACEEIEP are:

- ACEEIEP contains data, rather than a pointer.
- ACEEIEP contains a pointer, but the first word of the area pointed to by ACEEIEP does not contain the subpool and length information for the area.
- ACEEIEP contains a pointer, and the first word of the area pointed to contains the subpool and length information for a data area that points to additional area obtained using GETMAIN.

Note: If you currently use ACEEIEP in a nonstandard format and do not provide IRRACX01 and IRRACX02 exits, you might experience unpredictable results after the IRRACEE VLF class is activated. You might also experience unpredictable results if you have applications that use the ENVRIN and ENVROUT keywords on RACROUTE macros. Be sure to check whether applications you use do this.

Range tables

RACF gives control to IRRACX01 or IRRACX02 before compression of an ACEE if ACEEIEP is nonzero. This allows the exit to modify standard data before compression. More importantly, in the case of nonstandard data, the exit can build

and return a *range table* describing the data areas RACF is to compress when it compresses the ACEE. This description is necessary because nonstandard data, by definition, follows no set format for how non-contiguous data areas are hooked together. The range table allows you to tell RACF the start point and end point of multiple non-contiguous areas which need to be saved. The format of a range table is defined in *z/OS Security Server RACF Data Areas* under the ACXP mapping macro. Each data area pointed to by ACEEIEP, or by data areas chained off of ACEEIEP, should be included as a range in the range table (see "Range table example"). RACF moves the data specified in the range table to the contiguous area where it is stored. RACF also saves the range table in this contiguous area.

The range table is pointed to by X'8' into the exit parameter list. The exit should get the storage for the range table. When the exit provides a range table, RACF compression routines process the range table and ignore ACEEIEP. ICHRIX02 should FREEMAIN the data pointed by ACEEIEP.

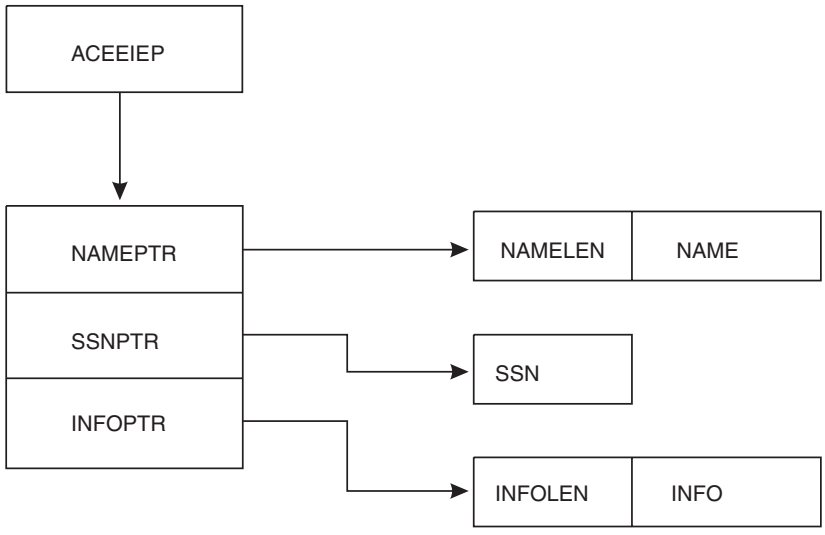
At expansion time, the process is reversed. The exit gets the storage for the original data structure, establishes appropriate pointers, and copies the data indicated by the range table into the storage.

At expansion time, RACF gives control to IRRACX01, but not to IRRACX02. As a result, when RACF's caller is in SRB mode or cross memory mode, an installation cannot count on a range table to restore nonstandard data. In these cases, you should use ACEEIEP data in standard format instead of a range table.

Range table example

Many nonstandard uses of ACEEIEP are possible. This example illustrates how to convert one possible nonstandard use into a range table.

Suppose you wish to compress an ACEE that is configured as follows:
where:



- NAMEPTR, SSNPTR, and INFOPTR are pointers to the actual data.
- NAME and NAMELEN are the full name and length of name of a person.
- SSN is a 9-character Social Security number.
- INFO and INFOLEN are miscellaneous data and the length of this data.

At compression time, the information pointed to by ACEEIEP must be converted into the following range table:

```
00000003                  * Range count
000000FF                  * Subpool

NAMEPTR+1                 * Pointer to start of first range
NAMEPTR+NAMELEN           * Pointer to end of first range

SSNPTR                    * Pointer to start of second range
SSNPTR+9                  * Pointer to end of second range

INFOPTR+1                 * Pointer to start of third range
INFOPTR+INFOLEN           * Pointer to end of third range
```

IRRACX01

Installing the exit routine

The exit routine must be located in the link-pack area: PLPA, FLPA, or MLPA. RACF initialization locates the exit at IPL time and places the exit address in the RACF communication vector table (RCVT). If RACF does not find the exit in the link-pack area, it sets the RCVT fields pointing to the exit to zero.

If you change the exit, you must re-IPL MVS for the changes to take effect.

Exit recovery

The exit should provide its own recovery. If the exit does not provide a recovery routine, the caller's recovery routine gets control.

Exit routine environment

The exit receives control in the following environment:

- In supervisor state with key 0.
- In AMODE(31). RACF does not validate the exit's addressing mode when loading it, and assumes that the exit can address parameters and data areas with 31-bit addresses.
- With no locks held.
- In non-cross-memory-mode.
- In task mode.

Exit routine processing

Compression-time invocation of IRRACX01: In non-cross-memory environments, IRRACX01 gets control before compression of an ACEE if ACEEIEP is nonzero. The compression function chooses which data (if any) is to be compressed with the ACEE, based on what IRRACX01 returns:

- If IRRACX01 returns no range table and if ACEEIEP is still nonzero, RACF assumes that it points to a standard ACEEIEP data area and compresses that data along with the ACEE.

  When the ACEE is expanded later, RACF expands this data automatically, without calling IRRACX01 again. IRRACX01 should be careful about making changes to ACEEIEP or the data it points to, as such changes affect the original, uncompressed ACEE.
- If IRRACX01 returns a range table, the compression function ignores the contents of the ACEEIEP field and compresses only the data indicated by the range table. IRRACX01 returns a range table by setting a pointer to it at offset X'8' into the exit parameter list.

Expansion-time invocation of IRRACX01: Expansion-time invocation of IRRACX01 occurs only if the IRRACX01 exit returned a range table at compression

time. This invocation allows IRRACX01 to properly rebuild the data structure, pointed to by ACEEIEP, from data passed to the exit by the range table.

The pointers in the range table point to the actual data that needs to be rebuilt into the data structure that existed before the ACEE was compressed. The saved range table is returned to the IRRACX01 exit, with updated data addresses.

RACF frees or reuses the range table and the referenced data areas in subsequent processing. The exit should not free them or change them.

The exit should copy the information that is pointed to by the range table into a data structure that the exit GETMAINs. (The logic to accomplish this should already be present, in most cases, in the ICHRIX01 exit that creates the information in the original ACEEIEP field.)

Note: MCS, in a sysplex configuration, can create a security environment using data from another system. ACEEXNVR is set on in the ACEE to indicate this. User exits should take the system of origin into consideration when processing the exit data.

When recreating ACEEIEP data in IRRACX01, you should realize that ICHRIX01 might not get control, and you might need to duplicate some of its function in IRRACX01. ("Fastpath" through RACROUTE does not go through ICHRIX01. See the description of RACROUTE REQUEST=VERIFY,SYSTEM=YES in *z/OS Security Server RACROUTE Macro Reference*.)

Programming considerations

Code the IRRACX01 exit routine to be reentrant and refreshable.

Link-edit IRRACX01 exit routines with AMODE(31) or AMODE(ANY) and with RMODE(24) or RMODE(ANY).

The exit must be aware that it can receive an ENVR object, or an ACEE created from an ENVR object, that originated on another system. If an ACEE was created from an ENVR object that originated on another system, the bit ACEEXNVR is set. If exits need to know the exact origin of the ACEE information, they can store this information in the installation data field pointed to by ACEEIEP.

Entry specifications

The system passes the address of the exit parameter list to the exit routine.

Registers at entry: The contents of the registers on entry to this exit are:

Register	Contents
0	Not applicable
1	Pointer to parameter list
2-12	Not applicable
13	Pointer to register save area
14	Return address
15	Entry point address of the exit routine

RACF uses the first word of the save area pointed to by register 13. The exit routine must not modify this part of the save area.

Parameter descriptions: Register 1 contains a pointer to the exit parameter list, ACXP, which is mapped by macro IRRACXP. See *z/OS Security Server RACF Data Areas* for a mapping of the ACXP data area.

Return specifications

Registers at exit: Upon return from this exit, the register contents must be:

Register	Contents
0-14	Restored to contents at entry
15	On the preprocessing call, the following return code:

Value	Meaning
0	Exit routine processing is complete.

Coded example of the exit routine
None.

IRRACX02

Installing the exit routine
The exit routine must be located in the link-pack area: PLPA, FLPA, or MLPA. RACF initialization locates the exit at IPL time and places the exit address in the RACF communication vector table (RCVT). If RACF does not find the exit in the link-pack area, it sets the RCVT fields pointing to the exit to zero.

If you change the exit, you must re-IPL MVS for the changes to take effect.

Exit recovery
The exit should provide its own functional recovery routine (FRR). If the exit does not provide an FRR, the FRR that RACF provides gets control.

Exit routine environment
The exit receives control in the following environment:

- In supervisor state with key 0.
- In AMODE(31). RACF does not validate the exit's addressing mode when loading it, and assumes that the exit can address parameters and data areas with 31-bit addresses.
- With no locks held.
- In cross-memory mode or SRB mode.

Exit routine processing

Compression-time invocation of IRRACX02: In cross-memory mode, IRRACX02 gets control before compression of an ACEE if ACEEIEP is nonzero. The compression function chooses which data (if any) is to be compressed with the ACEE, based on what IRRACX02 returns:

- If IRRACX02 returns no range table and if ACEEIEP is still nonzero, RACF assumes that it points to a standard ACEEIEP data area and compresses that data along with the ACEE.
- If IRRACX02 returns a range table, the compression function ignores the contents of the ACEEIEP field and compresses only the data indicated by the range table. IRRACX02 returns a range table by setting a pointer to it at offset X'8' into the exit parameter list.

Expansion-time invocation of IRRACX02: IRRACX02 is not invoked at expansion time.

Programming considerations
Code the IRRACX02 exit routine to be reentrant and refreshable.

Link-edit IRRACX02 exit routines with AMODE(31) or AMODE(ANY) and with RMODE(24) or RMODE(ANY).

The IRRACX02 exit receives control for cross-memory and SRB mode callers. It needs to be sensitive to the environment in which it is invoked and use only allowed services. If IRRACX02 changes ACEEIEP, the storage ACEEIEP points to must reside in the same address space as the ACEE (HOME). If IRRACX02 returns a range table, the range table must be in the primary address space, but the addresses in the range table must point to data in the HOME address space.

The exit must be aware that it can receive an ENVR object, or an ACEE created from an ENVR object, that originated on another system. If an ACEE was created from an ENVR object that originated on another system, the bit ACEEXNVR is set. If exits need to know the exact origin of the ACEE information, they can store this information in the installation data field pointed to by ACEEIEP.

Entry specifications

The system passes the address of the exit parameter list to the exit routine.

Registers at entry: The contents of the registers on entry to this exit are:

Register	Contents
0	Not applicable
1	Pointer to parameter list
2-12	Not applicable
13	Pointer to register save area
14	Return address
15	Entry point address of the exit routine

RACF uses the first word of the save area pointed to by register 13. The exit routine must not modify this part of the save area.

Parameter descriptions: Register 1 contains a pointer to the exit parameter list, ACXP, which is mapped by macro IRRACXP. See *z/OS Security Server RACF Data Areas* for a mapping of the ACXP data area.

Return specifications

Registers at exit: Upon return from this exit, the register contents must be:

Register	Contents
0-14	Restored to contents at entry
15	On the preprocessing call, the following return code: **Value** **Meaning** **0** Exit routine processing is complete.

Coded example of the exit routine

None.

Command exits for specific commands

There are two command exits, ICHCNX00 and ICHCCX00, that allow the installation to associate additional security checking or processing with certain RACF commands, or to bypass all security checking.

ICHCNX00 is called following syntax checking for:

- ADDSD command, before any authorization checking is performed.
- ALTDSD command, before the data set profile is retrieved.
- DELDSD command, before the data set profile is retrieved.
- LISTDSD command, before any data set profile is located for the ID, PREFIX, or DATASET parameters, to allow modification of the profile name to match RACF naming conventions, and after each data set profile is retrieved but before any authorization checking is performed.
- PERMIT command, before the data set profile is retrieved.
- SEARCH command, before the first data set profile is retrieved, to allow for modification of the profile name to match RACF naming conventions and after each data set profile is located but before any authorization checking is performed.
- IRRUT100 utility, after the data set profile is retrieved, but before the data set profile is associated with a user or group.
- IRRRXT00 (when RACROUTE REQUEST=EXTRACT is issued with CLASS=DATASET) before the data set profile is retrieved.

Note: The ALTDSD, DELDSD, LISTDSD, PERMIT, and SEARCH commands issue RACROUTE REQUEST=AUTH macros to check the command user's authority to a specified resource. The RACROUTE REQUEST=AUTH preprocessing and postprocessing exits therefore gain control from these commands. In addition, the ADDSD and DELDSD commands use the RACROUTE REQUEST=DEFINE macro to accomplish the data set definition, which means that the RACROUTE REQUEST=DEFINE preprocessing and postprocessing exits will gain control.

ICHCCX00 is called by the RACF commands DELUSER, DELGROUP, and REMOVE.

ICHCNX00 processing

The exit must be named ICHCNX00.

It allows an installation to perform additional security checks, to further enhance or restrict the RACF limitations on the passed commands, or to modify or eliminate the RACF DASD data set naming convention. Because corresponding processing might be required in the RACROUTE REQUEST=DEFINE preprocessing exit and the RACROUTE REQUEST=AUTH preprocessing or postprocessing exits, RACF passes these exits a parameter list with similar structure and content, to allow similar routines to be used.

RACF calls the naming conventions processing routine before ICHCNX00 receives control. See also "Data set naming convention table" on page 251.

This exit must be reentrant.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

This exit can run in the RACF subsystem address space, and considerations discussed in "Exits running in the RACF subsystem address space" on page 252 apply.

If the exit is invoked for a command that originates from a TSO user, it is invoked in problem state, under protection key 8, in an APF-authorized environment. If the exit

is invoked for a directed command, it is invoked in supervisor state, under protection key 0. If the exit is invoked for a command that originates from the operator's console, it is invoked in problem state, under protection key 2, in an APF-authorized environment. If the exit is invoked for a command issued under some other task, the invocation state depends on the attributes of that task.

z/OS Security Server RACF Data Areas contains a mapping of the command-preprocessing exit parameter list, CNXP.

The caller (indicated by the function and subfunction codes pointed to by the fullword at offset 4 in the parameter list) determines which parameters are passed to the exit routine and which parameters can be changed by the exit routine. See Table 14 for a summary of these parameters.

Table 14. ICHCNX00-exit parameter processing

CALLER		**OFFSET**											
		0	**4**	**8**	**12**	**16**	**20**	**24**	**28**	**32**	**36**	**40**	**44**
RACROUTE REQUEST=AUTH		P	P	P	C	0	C	0	P	C	0	0	0
RACROUTE REQUEST= DEFINE	DEFINE	P	P	P	C	0	C	0	P	C	C	0	0
	RENAME	P	P	P	C	C	C	0	P	C	C	0	0
	ADDVOL	P	P	P	C	0	C	C	P	C	0	0	0
	DELETE	P	P	P	C	0	C	0	P	C	0	0	0
ADDSD	SET	P	P	P	C	0	P[3]	0	P	C	C	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	C	0	P
ALTDSD	SET	P	P	P	C	0	P[3]	0	P	C	0	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	0	0	P
DELDSD	SET	P	P	P	C	0	P[3]	0	P	C	C	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	C	0	P
LISTDSD	Prelocate	P	P	P	C[1]	0	P	0	P	0	0	0	P
	DATASET	P	P	P	C	0	P	0	P	C	0	C	P
	ID or PREFIX	p	p	P	C	0	P	0	P	C	0	C	P
PERMIT	TO resource	P	P	P	C	0	P[3]	0	P	C	0	0	P
	FROM resource	P	P	P	C	0	P[4]	0	P	C	0	0	P
SEARCH	Presearch	P	P	P	C[2]	0	0	0	P	0	0	0	P
	Postsearch	P	P	P	C	0	P	0	P	C	0	0	P
IRRUT100		P	P	P	C	0	0	0	P	C	0	0	0
RACROUTE REQUEST= EXTRACT		P	P	P[5]	C	0	C[3]	0	P	C	P[5]	0	0

Table 14. ICHCNX00-exit parameter processing (continued)

CALLER	OFFSET											
	0	4	8	12	16	20	24	28	32	36	40	44

P means the field is passed to the exit routine, but should not be changed by the exit routine.

C means the field is passed to the exit routine, and can be changed by the exit routine.

0 means the field is not passed to the exit routine, and is indicated as zero.

Notes:

1. The field is set to the value specified (or defaulted to) on the DATASET, ID, or PREFIX parameter.
2. The field is set to the value specified on the MASK parameter, or to zero length if the NOMASK parameter was specified.
3. The field is nonzero only when the VOLUME parameter was specified.
4. The field is nonzero only when the FVOLUME parameter was specified. The address passed always points to zero.

Return codes from the command-preprocessing exit ICHCNX00

Except for a prelocate call to LISTDSD or SEARCH, when the ICHCNX00 preprocessing exit routine returns control, register 15 should contain one of the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Normal processing is to continue.
4	(4)	The request is not accepted, and is to be failed. The failure is to be logged (if logging is in effect), and a message is to be issued.
8	(8)	The request is not accepted, and is to be failed. The failure is to be logged (if logging is in effect), but no message is to be issued. Note, however, that messages can be issued through the PUTLINE I/O service routine by using the CPPL address passed at offset 44 in the parameter list. This return code allows the exit routine to fail the request, with the option of sending its own message without a RACF command message being issued.
C	(12)	Exit-routine processing is complete, and the request is granted. No authorization processing is to be performed, but other normal processing (such as logging) is to continue.

If register 15 contains any other value, processing proceeds as if the return code were 0.

Notes:

1. The prelocate call to ICHCNX00 from LISTDSD and SEARCH allows an installation to modify the name of the profile to be located so that it matches the naming conventions of RACF. RACF ignores the return code from a prelocate call. LISTDSD and SEARCH also issue a postlocate call to ICHCNX00. Therefore, you cannot use this exit to cancel a LISTDSD or SEARCH command until the postlocate call has been completed.
2. The data-set-type address, located at offset 36 in the parameter list, is zero except as a result of ADDSD, RACROUTE REQUEST=DEFINE DEFINE, and

RACROUTE REQUEST=DEFINE RENAME processing. In these cases, the exit can set the field to be used by the caller to determine whether the data set to be created is a user data set or a group data set.

3. Only return codes 0 and 4 are valid for RACROUTE REQUEST=EXTRACT.

When return codes 0 and C are issued for ADDSD, RACROUTE REQUEST=DEFINE DEFINE, and RACROUTE REQUEST=DEFINE RENAME, the exit must supply sufficient information to allow RACF to determine the type of data set to be created.

When the exit return code is 0:

- If the data set type is set to X'80', a user profile must exist to match the qualifier field (at offset 32).
- If the data set type is set to X'40', a group profile must exist to match the qualifier field (at offset 32).
- If the data set type is set to X'01' or to any other value, either a user or a group profile must exist.

In each of the above cases, normal authorization processing continues.

When the exit return code is C:

- If the data set type is set to X'80' or X'40', the request is processed.
- If the data set type is set to X'01' or to any other value, either a user or a group profile must exist, but the command issuer need not have any other authority.

ICHCCX00 processing

The exit must be named ICHCCX00. It is entered after syntax checking and before any data set profile is located.

The ICHCCX00 exit allows an installation to perform additional security checks and to further enhance or restrict the RACF limitations on the passed commands.

This exit must be reentrant.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

This exit can run in the RACF subsystem address space, and considerations discussed in “Exits running in the RACF subsystem address space” on page 252 apply.

If the exit is invoked for a command that originates from a TSO user, it is invoked in problem state, under protection key 8, in an APF-authorized environment. If the exit is invoked for a directed command, it is invoked in supervisor state, under protection key 0. If the exit is invoked for a command that originates from the operator’s console, it is invoked in problem state, under protection key 2, in an APF-authorized environment. If the exit is invoked for a command issued under some other task, the invocation state depends on the attributes of that task.

z/OS Security Server RACF Data Areas contains a mapping of the command-preprocessing exit parameter list, CCXP.

Return codes from the command-preprocessing exit ICHCCX00

When the ICHCCX00 preprocessing exit routine returns control, register 15 should contain one of the following return codes:

Code	Meaning
0	Exit-routine processing is complete. Normal processing is to continue.
4	The data set search is to be bypassed.
8	The request is failed, and a message is issued.

Note: If register 15 contains any other value, processing proceeds as if the return code were 0.

Common command exit

The IRREVX01 exit provides a common exit point for most RACF commands. The exit gets control before and after the execution of all RACF commands except:

- BLKUPD
- RACDCERT
- RACLINK
- RVARY
- Commands that cannot be issued from TSO, such as DISPLAY, RESTART, SET, SIGNOFF, and STOP. (For information on whether a command can be issued from TSO, see *z/OS Security Server RACF Command Language Reference*.)

For a list of the commands for which the exit gets control, and the code that is passed for each command, see the mapping of the EVXP data area in *z/OS Security Server RACF Data Areas*.

Using the information it receives about the command and the command issuer, the exit routine can:

- Modify a command before it executes
- Prevent a command from executing

Controlling the exit routine through the dynamic exits facility

IBM has defined the IRREVX01 exit point to the dynamic exits facility. Therefore, you can update the exit without re-IPLing. You can associate your installation exit routine with the IRREVX01 exit point via any of the following:

- The PROG*xx* member of SYS1.PARMLIB
- The SETPROG EXIT operator command
- An authorized program issuing the CSVDYNEX macro

For example, to add load module IRREVX1A to the IRREVX01 exit point, add the following to the PROG*xx* member:

```
EXIT  ADD
      EXITNAME(IRREVX01)
      MODNAME(IRREVX1A)
      STATE(ACTIVE)
```

Alternatively, from the console issue the command:

```
SETPROG EXIT,ADD,EXITNAME=IRREVX01,MODNAME=IRREVX1A
```

For more information on the dynamic exits facility, see *z/OS MVS Installation Exits*. For information on the PROG*xx* member of SYS1.PARMLIB, see *z/OS MVS Initialization and Tuning Reference*. For information on the SETPROG EXIT command, see *z/OS MVS System Commands*. For information on the CSVDYNEX macro, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Replacing the exit routine

For information on replacing a dynamic exit routine, see *z/OS MVS Installation Exits*.

Be careful if you replace the exit with a new one that is not compatible with the old one; for example, if the new one does not clean up fields that the old one set. Because the exit gets control before and after a command executes, in such cases you should not replace the exit while commands are executing.

Exit routine environment

The exit receives control in the following environment:

- Enabled for interrupts
- In supervisor state with key 0
- In AMODE(31) and RMODE(ANY)
- With no locks held
- Not in cross-memory mode
- Not in ASC mode
- In either the command issuer's address space or the RACF subsystem address space

Exit recovery

Each exit routine must provide its own recovery routine, which gets control if the exit routine abends. Note that if there are multiple exit routines at a given exit point and one of those routines abends, control is not given to the remaining exit routines or their recovery routines. You should take this fact into consideration when designing the recovery scheme for the IRREVX01 exit point.

Exit routine processing

RACF gives control to IRREVX01 both before and after a command is executed. The exit is given control regardless of the command's source (for example, TSO session, operator console, RACF parameter library, application program, or exit).

For commands that specify the AT or ONLYAT keyword, RACF invokes the exit on each target system on which the command executes and the exit exists.

If automatic direction is active, RACF invokes the exit before and after the command executes on the local system. RACF then sends the command to the remote systems. If an IRREVX01 exit exists on a remote system, RACF invokes that exit before and after the command executes on the remote system.

Information passed in the parameter list

The parameter list passed to the exit contains:

- A flag indicating whether the exit is executing in the RACF subsystem address space or the command issuer's address space.
- A function code that identifies the command name. If the exit changes this function code, the change has no effect on RACF's processing of the command.
- A flag indicating whether this is the preprocessing or postprocessing call.
- Flags indicating whether the command was directed to the node and if so, how—with the AT or ONLYAT keywords or by automatic command direction.
- A pointer to a command buffer that contains:
  - An image of the original command after parsing.

- If the AT or ONLYAT keyword was specified, it does not appear in the command buffer.
- All general resource names appear in the appropriate case for the class. For classes specified with CASE=ASIS in the class descriptor table, such as EJBROLE and GEJBROLE, the case is as entered by the user. For all other classes, profile names appear in upper case.
- If the user's TSO profile is not set to NOPREFIX, all unquoted data set names entered by the user are prefixed with the user's user ID and enclosed within single quotes.
- For the RDEFINE, RALTER, RLIST, RDELETE, PERMIT, and ADDSD commands, if a class name was abbreviated in the command, the full name of the class appears in the command image.

  Exception: A profile name in the GLOBAL class (which is a class name) is left as it was entered in the command.

– The defaults for command keywords that have defaults and were not specified on the original command.
– Any data that was provided via prompting.
– An additional 300 bytes of blanks following the last keyword in the buffer, where the exit can add additional keywords.

The exit can change the values of keywords in the buffer, but if it changes the command name RACF fails the command. If the exit changes the pointer to the command buffer, RACF ignores the change.

- The address of an ACEE:
  – If the address is 0, the RACF parameter library issued the command, and the command runs with the authority of the RACF subsystem address space.
  – If the address is nonzero, it points to the ACEE of the user ID under whose authority the command runs. (This user ID is usually the one that issued the command, but not necessarily. For example, for directed commands it is the user ID specified on the AT or ONLYAT keyword.) The exit can examine the user ID and group name in this ACEE to do authority checking on the command. The exit can modify fields in the ACEE that are part of the defined programming interface, but the postprocessing call should restore the previous values when it gets control after command execution or after an abend. For information on the ACEE fields, see *z/OS Security Server RACF Data Areas*.

  The exit cannot change the pointer to the ACEE.
- The originating node and user ID, if the command was directed with the AT or ONLYAT keyword, or with automatic command direction. The exit can use these values to make decisions, but cannot change them.
- A pointer to a word that the exit can use to communicate between the preprocessing call and the postprocessing call to an exit routine, or between different exit routines associated with the exit. The exit can change the contents of this communication area, but not the pointer to it.
- A command return code, an abend completion code, and a flag indicating whether the command abended:
  – On the preprocessing call, these values are 0. If the exit changes these values, the changes are ignored.
  – On the postprocessing call:
    - If the command did not abend, the command return code field contains the value set by the command processor during command execution. The exit can change this return code.

  - If the command abended, the flag is set to indicate that the abend has occurred, the abend completion code is passed, and the command return code field is set to the abend reason code, if available. If the exit changes these values, the changes are ignored.

- A pointer to a message area. If the exit fails the command with a message, it can provide message text in this area to be inserted into message IRRV022I. The exit cannot change the pointer.

The preprocessing call

Before RACF makes the preprocessing call to IRREVX01, it parses the command to ensure that it is syntactically correct. If RACF cannot successfully parse the command, it does not give control to the exit. If the command was issued as a RACF operator command, RACF verifies that the user is allowed to issue the command as an operator command based on the OPERCMDS authority check. If the user does not have the required authorization, RACF does not give control to the exit.

RACF does not process the naming convention table before making the preprocessing call. Therefore, data set names in the command buffer might be changed later by the naming conventions.

For commands that specify the AUTOUID or AUTOGID keyword, the command image contains only the AUTOUID or AUTOGID keyword. The derived UID or GID value is contained in the command image that is sent to the postprocessing call.

Based on the information passed to it in the parameter list, the exit routine can:

- Make changes to the command before the command executes, by updating the command image passed to it.
- Determine whether the command executes or fails, by setting a return code in register 15, as described in “Registers at exit” on page 271. If the routine returns a nonzero return code, the command fails with a return code of 8.
- Determine whether RACF issues message IRRV022I for a failed command, by setting a return code in register 15, as described in “Registers at exit” on page 271. The exit routine has the option to provide text to be appended to message IRRV022I.

If the exit routine makes a change to the command buffer in the preprocessing call, RACF parses the command again. RACF does this parse in noprompt mode, to prevent the user from entering values or adding keywords that the exit would find unacceptable. If this parse fails, or if RACF finds that the exit routine changed the command name, RACF fails the command and calls the exit routine for the postprocessing call, to allow the exit to clean up and reset values set by the preprocessing call.

The postprocessing call

After the preprocessing call completes and RACF optionally reparses the command, RACF processes the command. The processing of the command includes processing the naming convention table. RACF then makes the postprocessing call to the exit. If the exit changed the command buffer in the preprocessing call, RACF passes the re-parsed command image to the postprocessing call.

For commands that specify the AUTOUID or AUTOGID keyword, the command image in the postprocessing call contains both the AUTOUID or AUTOGID keyword, and the UID or GID keyword with the value RACF has derived for the UID or GID. For example, if the user entered:

```
ADDUSER MARTIN OMVS(AUTOUID HOME(/u/martin) PROGRAM(/bin/sh))
```

and RACF derived a UID value of 907, the postprocessing exit would see:

```
ADDUSER MARTIN OMVS(AUTOUID UID(907) HOME(/u/martin) PROGRAM(/bin/sh))
```

RACF makes the postprocessing call regardless of the return code from the preprocessing call. The exit receives the same parameter list on the postprocessing call as on the preprocessing call. The postprocessing call can alter the ACEE and the communications area passed to it, and do any cleanup required due to changes made in the preprocessing call.

If the command did not abend, the command return code field contains the return code set by the command processor during command execution. The command can change this return code, and if it does, the changed value is returned to the command issuer. However, if automatic command direction is active, RACF uses the original return code to determine whether to automatically direct the command.

If the command abended, the postprocessing call receives a flag indicating that the command abended. The exit routine should do any cleanup required due to changes made in the preprocessing call, just as it would if the command had completed without an abend.

Programming considerations

Code the IRREVX01 exit routine to be reentrant.

Link-edit IRREVX01 exit routines with AMODE(31) or AMODE(ANY) and with RMODE(ANY).

Entry specifications

The system passes the address of the exit parameter list to the exit routine.

Registers at entry

The contents of the registers on entry to this exit are:

Register	Contents
0	Not applicable
1	Pointer to parameter list
2-12	Not applicable
13	Pointer to register save area
14	Return address
15	Entry point address of the exit routine

RACF uses the first word of the save area pointed to by register 13. The exit routine must not modify this part of the save area.

Parameter descriptions

Register 1 contains a pointer to the exit parameter list, EVXP, which is mapped by macro IRREVXP. See *z/OS Security Server RACF Data Areas* for a mapping of the EVXP data area.

Return specifications

On the preprocessing call the exit routine passes back a return code indicating whether processing of the command should continue or stop. The return code from the exit is the highest return code from all active exit routines for the IRREVX01 exit point.

On the postprocessing call RACF ignores any return code from the exit. The exit should pass back a return code of 0.

Registers at exit

Upon return from this exit, the register contents must be:

Register	Contents
0-14	Restored to contents at entry
15	On the preprocessing call, one of the following return codes: **Value** **Meaning** **0** Continue processing the command. If the exit has changed the command buffer, RACF reparses the command. **4** The exit has failed the command, and RACF is not to issue a message. **8** The exit has failed the command, and RACF is to issue a message to the user. The message indicates that the exit has failed the command, and might contain additional text returned by the exit. On the postprocessing call, 0

Coded example of the exit routine

The RACEXITS member in SYS1.SAMPLIB includes two sample IRREVX01 exit routines, IRREVX1A and IRREVX1B.

IRREVX1A illustrates how to use the IRREVX01 exit point to fail certain commands.

IRREVX1B illustrates how to use the IRREVX01 exit point to limit SPECIAL authority for certain user IDs to updating password information. The exit checks whether a FACILITY class profile of the form HELPDESK.*userid* exists, and if so, limits the SPECIAL authority to password updates. Note however, that generic profile checking applies to this profile lookup. If your installation already uses the FACILITY class, before you activate the IRREVX1B routine to the IRREVX01 exit point make sure that no profile such as ** exists. Otherwise, no user IDs will have SPECIAL authority until you deactivate the exit routine.

New-password exit

RACROUTE REQUEST=VERIFY processing and the ALTUSER and PASSWORD commands invoke the installation-supplied new-password processing exit.

The installation has the option of using this exit to augment RACF function when establishing a new password or a new password interval.

This exit can examine the intended new password and the new password-change interval (if invoked from the PASSWORD command). In the case of new-password processing, the exit unconditionally gains control whenever a new password is specified.

In a remote sharing environment, if password synchronization or automatic password direction is active, and a password is changed, the new-password exit is always invoked on the node where the initial password change is made. When RACF automatically updates the password on other nodes, the new-password exit might or might not be invoked:

- If the password was changed by a RACF command, and the command is propagated to another node by automatic command direction, the new-password exit is invoked on that node.
- If the password was changed by other means (at logon, or by a RACROUTE or ICHEINTY invocation), and the password change is propagated to another node by automatic password direction or password synchronization, the new-password exit is not invoked on that node.

ICHPWX01 processing

The new-password exit must be named ICHPWX01.

This exit can run in the RACF subsystem address space, and considerations discussed in “Exits running in the RACF subsystem address space” on page 252 apply.

This exit must be reentrant. It can have any RMODE but should use AMODE(31) or AMODE(ANY) as the AMODE for the best use of virtual storage and best RACF performance.

When called from RACROUTE REQUEST=VERIFY processing, this exit is invoked in supervisor state, under protection key 0.

When called from the ALTUSER or PASSWORD commands:

- If the command originates from a TSO user, the exit is invoked in problem state, under protection key 8, in an APF-authorized environment.
- If the command is a directed command, the exit is invoked in supervisor state, under protection key 0.
- If the command originates from the operator’s console, the exit is invoked in problem state, under protection key 2, in an APF-authorized environment.
- If the command was issued under another task, the invocation state depends on the attributes of that task.

z/OS Security Server RACF Data Areas contains a mapping of the exit parameter list, PWXP.

The ICHPWX01 routine is invoked in the following ways:

- **Through RACROUTE REQUEST=VERIFY processing.** If you specify a new password, REQUEST=VERIFY performs the following functions:
  1. Invokes ICHRIX01 (if ICHRIX01 is present in the system)
  2. Validates the new password for correct alphanumeric syntax and compliance with the installation’s syntax rules
  3. Invokes ICHPWX01 (if ICHPWX01 is present in the system)
- **Through the ALTUSER command.** After parsing and checking the user’s authorization:
  - If you specify the PASSWORD keyword with NOEXPIRED, ALTUSER validates the new password against the installation’s syntax rules and invokes ICHPWX01.
  - If you specify the PASSWORD keyword with a password value and do not specify NOEXPIRED, ALTUSER invokes ICHPWX01. The syntax rules do not apply. The user is required to change the password at the next logon or start of a job.

  - If you specify the PASSWORD operand without a value and do not specify NOEXPIRED, the password defaults to that of the user's default group. In that case, ICHPWX01 is not invoked. The user is required to change the password at the next logon or start of a job.
- **Through the PASSWORD command.** If you specify the PASSWORD or INTERVAL keywords and the conditions listed below are met, PASSWORD invokes ICHPWX01 after parsing and checking the user's authorization:
  - The new password differs from the current password.
  - The new password differs from the previous passwords, if the password-history option is active.
  - The new password obeys all of the installation's syntax rules.

Table 15 shows which fields are available to the exit when the exit is called from the different RACF components.

Table 15. Fields available during ICHPWX01 processing

OFFSET (Decimal)	PARAMETER (Address)	REQUEST= VERIFY	ALTUSER	PASSWORD
0	Length	X	X	X
4	Caller	X	X	X
8	Command-processor parameter list	—	X	X
12	NEWPASS	X	X	O
16	INTERVAL	—	—	O
20	User ID	X	X	X
24	Work area	X	X	X
28	Current password	X[1]	—	O[2]
32	Password last change date	X	—	O[2]
36	ACEE	X[3]	X	X
40	Group name	O	—	—
44	Installation data	O	—	—
48	Password history	X	—	O[2]
52	Flag byte	X	—	—

X means "always available."
O means "might be available."
— means "never available."

Notes:
1. Not available if PASSCHK=NO was specified.
2. Available only if NEWPASS is available.
3. Although available, the ACEE might not be fully initialized.

Return codes from the new-password exit

When the password exit routine returns control, register 15 should contain one of the following return codes:

Hex	(Decimal)	Meaning
0	(0)	The new password field and the interval value are copied back into the calling function. Continue with processing.

Hex	(Decimal)	Meaning
4	(4)	The new password request is not accepted and is to be failed. RACROUTE REQUEST=VERIFY processing terminates with a return code indicating a new password that is not valid. The ALTUSER command ignores the request and continues processing. The PASSWORD command terminates processing.
8	(8)	The interval-value-change request is not accepted and is to be failed. The PASSWORD command will terminate processing.
C	(12)	The new password request is not accepted and is to be failed. This return code is the same as return code 4, except that error messages issued by the ALTUSER and PASSWORD commands are suppressed if the exit itself has already issued an appropriate message.
10	(16)	The request to change the interval value is not accepted and is to be failed. This return code is the same as return code 8 except that error messages issued by the ALTUSER and PASSWORD commands are suppressed if the exit itself has already issued an appropriate message.

Note: Decimal return codes 0 and 4 are valid for RACROUTE REQUEST=VERIFY; return codes 0, 4, 12, and 16 are valid for ALTUSER, and 0, 4, 8, 12, and 16 are valid for PASSWORD. If register 15 contains any other values, processing ends with an abend.

Possible use of the exit

Password quality control

One of the main objections to the use of passwords generated and maintained by the user is that the passwords chosen might readily be guessed. User education is one way to try to resolve the problem. An alternative is to use the system to ensure that the passwords selected are suitable.

Whenever a user enters the system, RACF invokes the RACROUTE REQUEST=VERIFY function. At this time the user is able to (or might be forced to) change passwords. The installation can devise whatever tests it wishes to ensure that the password supplied meets the required standard.

RACF gives you the ability to specify password-content rules with the SETROPTS command. You can make additional checks, using the exit routines. Because the new-password exit is called by both REQUEST=VERIFY and the PASSWORD command, this exit is a good place to make the additional checks on new passwords.

For example with the SETROPTS command, you can ensure that the password is more than six characters or that it contains an alphanumeric mix. With an exit, more complex tests can disallow names, months, user IDs, and group names, or detect trivial usage of alphanumeric mixes such as JAN98 and FEB01.

The use of the new-password exit augments the installation's syntax rules. Be sure that the exit and the syntax rules do not contradict each other. For example, if the

installation requires that passwords contain all numerics and the exit requires an alphabetic character in the password, you cannot create a new password.

Password authentication exits

There are two password authentication exit routines, ICHDEX01 and ICHDEX11. The RACF manager calls ICHDEX01 whenever it is necessary to store or compare encrypted password or OIDCARD data in a user profile. RACROUTE REQUEST=EXTRACT processing calls ICHDEX01 when TYPE=ENCRYPT,ENCRYPT=(...,INST) is specified for BRANCH=NO. When BRANCH=YES is specified, RACROUTE processing calls ICHDEX11. ICHDEX01 and ICHDEX11 perform equivalent function.

These exits enable an installation to do the following:

- Use its own authentication algorithm
- Use only the masking algorithm to perform encoding
- Use only the RACF DES algorithm to perform authentication (see “Migrating to the RACF DES authentication algorithm” on page 54 for more information)

See “Password authentication options” on page 54 for more information on password authentication options.

To use an installation-provided method of user verification, set the return code in the ICHDEX01 exit to 0. As a result, RACF uses the encoding routine coded in the exit. You should also provide an ICHDEX11 exit to perform the same function.

To use the masking algorithm as the only means of logon checking, set the return code in the ICHDEX01 exit to 4. You should also provide an ICHDEX11 exit that sets the same return code.

To use only the RACF DES algorithm for checking user IDs, set the return code in the ICHDEX01 exit to 8. You should also provide an ICHDEX11 exit that sets the same return code. This might be the method you want to use if your installation is a new user of RACF and has never used the masking algorithm.

If you do not provide an ICHDEX01 exit and activate it as described in “Installing the exit routine” on page 276, RACF uses the two-step method of checking described in “Migrating to the RACF DES authentication algorithm” on page 54. When you install the RACF component of the Security Server, the ICHDEX01 exit is not active and the two-step method of checking is used.

When using the two-step method of checking, there is an extremely remote possibility that the *RACF DES-encoded* form of one user’s password is identical to the *masked* form of another user’s password. As long as your installation uses the two-step method of checking, your installation might have an exposure. To avoid this possibility, after all the users at your installation have been RACF DES-encoded using the two-step verification and conversion process, provide an ICHDEX01 exit that sets the return code to 8. This return code directs RACF to use only the RACF DES algorithm for logon checking.You should also provide an ICHDEX11 exit that sets the same return code.

RACF provides a version of ICHDEX01 that unconditionally returns with a return code of 4 to force RACF to use the masking algorithm. The RACF-provided version

of ICHDEX01 is shipped in SYS1.LINKLIB, where it is not found during initialization. As a result, the DES algorithm, using the two-step method of checking, is the default.

If you use the RACF-provided version of ICHDEX01, you can also use it as the ICHDEX11 exit. You must create the appropriate module in the link pack area.

ICHDEX01

Installing the exit routine

IBM provides an ICHDEX01 exit in SYS1.LINKLIB that causes RACF to use the masking algorithm to authenticate passwords. To use the ICHDEX01 exit that IBM provides, you must activate it by installing it in the link pack area so that RACF finds it during initialization. There are two methods you can use:

- Use an IEALPA*xx* member in SYS1.PARMLIB to request that MVS load ICHDEX01 from SYS1.LINKLIB as a temporary extension to the existing link pack area. Modify all your IEASYS*xx* members to specify that MVS should use this IEALPA*xx* member. See *z/OS MVS Initialization and Tuning Guide* for information. See member RACPARM in SYS1.SAMPLIB for a sample IEALPA*xx* member.
- Create an SMP/E USERMOD to move ICHDEX01 into LPALIB.

To install an installation-provided ICHDEX01 exit, name the exit ICHDEX01 and load it into the link pack area (LPA).

Exit recovery

The exit should provide its own recovery, as an ESTAE.

Exit routine environment

The exit receives control in the following environment:

- In supervisor state
- Under protection key 0
- With no locks held

Exit routine processing

The ICHDEX01 exit is called by the RACF manager whenever it is necessary to store or compare encrypted password or OIDCARD data in a user profile. The exit is also called by RACROUTE REQUEST=EXTRACT processing when TYPE=ENCRYPT,ENCRYPT=(...,INST) is specified for BRANCH=NO.

Programming considerations

This exit must be reentrant.

RACF might have enqueued on the RACF database containing the user profile (either a shared or exclusive enqueue) and might have RESERVEd the DASD volume on which it is located. The exit can not issue any RACF macros or call the RACF manager.

The exit can have any RMODE, but should be AMODE(31).

Entry specifications

The system passes the address of the exit parameter list to the exit routine.

Registers at entry: The contents of the registers on entry to this exit are:

Register **Contents**

0	Not applicable
1	Pointer to parameter list
2-12	Not applicable
13	Pointer to register save area
14	Return address
15	Entry point address of the exit routine

RACF uses the first word of the save area pointed to by register 13. The exit routine must not modify this part of the save area.

Parameter descriptions: Register 1 contains a pointer to the exit parameter list, DEXP, which is mapped by macro IRRDEXP. See *z/OS Security Server RACF Data Areas* for a mapping of the DEXP data area.

Return specifications

Registers at exit: Upon return from this exit, the register contents must be:

Register	Contents
0-14	Restored to contents at entry
15	For an encrypt operation, one of the following return codes:

Value	Meaning
0	The exit has encrypted the data and placed the results in the area pointed to by the address at offset 16 (X'10') in the parameter list. The length of the encrypted data must be the same as that of the clear text data.
4	The exit has not encrypted the data. RACF is to encrypt the data, using the masking algorithm.
8	The exit has not encrypted the data. RACF is to encrypt the data, using the RACF DES algorithm.
16	The exit has not encrypted the data. RACF is to encrypt the data, using the RACF DES algorithm.

For a compare operation, one of the following return codes:

Value	Meaning
0	The clear text data and the encrypted data should be considered equal.
4	RACF is to attempt to compare the data by using the masking algorithm.
8	RACF is to attempt to compare the data by using the RACF DES algorithm.
12	The clear text data and the encrypted data should be considered unequal.
16	RACF is to attempt to compare the data by using the RACF DES algorithm. If DES processing fails, RACF uses masking.

Note: If register 15 contains any other value, RACF treats it as a return code of 4.

Coded example of the exit routine

None.

ICHDEX11

Installing the exit routine

There are two methods you can use to install your ICHDEX11 exit:

- Use an IEALPA*xx* member in SYS1.PARMLIB to request that MVS load ICHDEX11 from your library as a temporary extension to the existing link pack area. Modify all your IEASYS*xx* members to specify that MVS should use this IEALPA*xx* member. See *z/OS MVS Initialization and Tuning Guide* for information.
- Create an SMP/E USERMOD to create ICHDEX11 in LPALIB.

Exit recovery

The exit should provide its own functional recovery routine (FRR). If the exit does not provide an FRR, the FRR that RACF provides gets control.

Exit routine environment

The exit receives control in the following environment:

- In supervisor state
- Under protection key 0
- With no locks held
- From a branch-entered service.
- In task mode with an EUT FRR, or SRB mode

Exit routine processing

The ICHDEX11 exit is called by RACROUTE REQUEST=EXTRACT processing when TYPE=ENCRYPT, ENCRYPT=(...,INST) is specified for BRANCH=YES. It performs the same function that the ICHDEX01 exit performs.

Programming considerations

This exit must be reentrant.

The exit must execute in AMODE(31) to access some parameters.

The exit cannot issue SVCs.

Entry specifications

The system passes the address of the exit parameter list to the exit routine.

Registers at entry: The contents of the registers on entry to this exit are:

Register	Contents
0	Not applicable
1	Pointer to parameter list
2-12	Not applicable
13	Pointer to register save area
14	Return address
15	Entry point address of the exit routine

RACF uses the first word of the save area pointed to by register 13. The exit routine must not modify this part of the save area.

Parameter descriptions: Register 1 contains a pointer to the exit parameter list, DEXP, which is mapped by macro IRRDEXP. See *z/OS Security Server RACF Data Areas* for a mapping of the DEXP data area.

Return specifications

Registers at exit: Upon return from this exit, the register contents must be:

Register	Contents
0-14	Restored to contents at entry
15	For an encrypt operation, one of the following return codes:

Value	Meaning
0	The exit has encrypted the data and placed the results in the area pointed to by the address at offset 16 (X'10') in the parameter list. The length of the encrypted data must be the same as that of the clear text data.
4	The exit has not encrypted the data. RACF is to encrypt the data, using the masking algorithm.
8	The exit has not encrypted the data. RACF is to encrypt the data, using the RACF DES algorithm.
16	The exit has not encrypted the data. RACF is to encrypt the data, using the RACF DES algorithm.

Note: If register 15 contains any other value, RACF treats it as a return code of 4.

Coded example of the exit routine

None.

RACROUTE REQUEST=AUTH exits

A RACROUTE REQUEST=AUTH determines whether a user is authorized to obtain use of a resource (such as a DASD data set, or any resource defined by classes in the class-descriptor table) protected by RACF. When a user requests access to a RACF-protected resource, RACROUTE REQUEST=AUTH bases acceptance of the request on the identity of the user and whether the user has been permitted sufficient access authority to the resource.

You can use the RACROUTE REQUEST=AUTH exit routine to perform additional authorization checks for users or to modify the logging option for access to a resource. (Logging can be suppressed or requested when accessing a specified resource.) For example, resource managers, such as catalog management, can use RACROUTE REQUEST=AUTH to determine whether a resource (including DATASET) is RACF-protected.

Note: If the request is for the OPERCMDS class, authority checking for critical system commands might be in process. If the request is for the FIELD class, RACF might have already obtained serialization. In either case, no additional processing that accesses the RACF database or halts the completion of processing should be done. Otherwise, the system might hang.

Many of the values passed to the exits are derived from the parameters specified on the macro. For details of the RACROUTE REQUEST=AUTH macro, see *z/OS Security Server RACROUTE Macro Reference*.

Extended addressing

In many cases, RACF must copy caller-supplied parameter areas to an area below 16MB, so that the exit can address the parameter areas. However, when the RACROUTE REQUEST=AUTH exit routines receive the ACCLVL parameter or INSTLN parameters, RACF does not know the format or length of the parameters

being passed, and therefore cannot copy the parameters into 24-bit storage. Because the parameter list pointing to these parameters is in 24-bit storage, you must modify the exit routines to handle the parameters if the exit routines access these areas, and if they are passed by callers in 31-bit mode.

Preprocessing exit (ICHRCX01)

The RACROUTE REQUEST=AUTH preprocessing exit routine must be named ICHRCX01.

This exit must be reentrant and is invoked in supervisor state, with protection key 0, with no locks held. The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

When the RACROUTE REQUEST=AUTH preprocessing exit receives control for the DATASET class, RACF has already processed the naming convention table, if there is one, and if the profile name was changed by the table then this exit is passed the modified profile name.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=AUTH exit parameter list, RCXP.

Return codes from the RACROUTE REQUEST=AUTH preprocessing exit

When the RACROUTE REQUEST=AUTH preprocessing exit routine returns control, register 15 should contain one of the following return codes. Do not confuse these return codes with the return codes from the RACROUTE REQUEST=AUTH macro, the meanings of which are documented in *z/OS Security Server RACROUTE Macro Reference*.

When the RACROUTE REQUEST=AUTH preprocessing exit returns a return code of 4 or 8 and the RACROUTE REQUEST=AUTH macro specified ENTITY=(entity address, CSA) or a private-area profile (see flag byte 3), the exit routine must create a profile and return the address of the profile in Register 1. The first word in the profile must contain the subpool number and the length of the profile.

Hex	(Decimal)	Meaning
0	(0)	Exit-routine processing is complete. Normal processing is to continue.
4	(4)	The request is not accepted and is to be failed; however, the postprocessing exit is still invoked.
8	(8)	The request is accepted. No more processing is performed; however, the postprocessing exit is still invoked.
C	(12)	Exit-routine processing is complete and the request is to be granted. RACROUTE REQUEST=AUTH is not to perform any authorization checking on the access list, but other normal REQUEST=AUTH processing (such as default return code processing, PROTECTALL processing, and logging) is to continue.

Notes:

1. If register 15 contains any other value, RACROUTE REQUEST=AUTH issues an abend code (382) that indicates a non-valid exit return code.
2. The RACROUTE REQUEST=AUTH exit parameter list points to the naming-convention parameter list. For a description of what happens if you

change the naming-convention parameter list when you code the REQUEST=AUTH preprocessing exit, see the description of the naming-convention exit, CNXP, in *z/OS Security Server RACF Data Areas*.

RACF uses resident profiles in two ways:
- As installation-supplied profiles
- As specified by an exit routine

The ICHRRPF macro maps the resident profile. *z/OS Security Server RACF Data Areas* contains a mapping of RRPF.

If a profile is created that does not conform to the standard format, it is the responsibility of the RACROUTE REQUEST=AUTH preprocessing exit routine to ensure that RACF does not refer to that profile (that is, do not specify an exit return code of 0 if a subsequent RACROUTE REQUEST=AUTH is issued specifying the profile you built as input via the PROFILE keyword). Note, however, that RACF's caller can also examine the profile, so you should build one that has appropriate data in it or the results will be unpredictable.

Postprocessing exit (ICHRCX02)

The RACROUTE REQUEST=AUTH postprocessing exit routine must be named ICHRCX02.

This exit must be reentrant and is invoked in supervisor state, with protection key 0, with no locks held. The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

When the RACROUTE REQUEST=AUTH postprocessing exit routine receives control, RACF has already performed the main function (for example, authorization checking), but has not performed any logging or statistics recording.RACF has also processed the naming convention table, if there is one. If the profile name was changed by the table, this exit is passed the modified profile name.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=AUTH exit parameter list, RCXP.

In some cases, the RACF return code passed to the exit (and addressed by RCXRCODE) is changed by RACF before it is returned to the caller of RACROUTE REQUEST=AUTH. These cases include:
- If PROTECTALL (FAILURES) is active and a data set profile is not found, a return code of 4 is passed when the exit is called. However, if the user ID does not have SPECIAL authority over the data set name, the final RACF return code is 8, not 4.
- If the return code passed to the exit is 4, but the default return code for the class is not 4, the final RACF return code is the default return code for the class.
- If the return code passed to the exit is 4, and RACFIND=YES was specified because the data set was RACF-indicated, the final RACF return code is 8.
- If the user ID in the ACEE or in the TOKEN is *BYPASS*, the final RACF return code is 4.
- If STATUS=ACCESS was requested, the final RACF return code is 20.

Return codes from the RACROUTE REQUEST=AUTH postprocessing exit

When the RACROUTE REQUEST=AUTH postprocessing exit routine returns control, register 15 should contain one of the following return codes. Do not confuse these return codes with the return code from the RACROUTE REQUEST=AUTH service, the meanings of which are documented in *z/OS Security Server RACROUTE Macro Reference*.

Code	Meaning
0	Continue with REQUEST=AUTH processing. (If the exit routine changes the return or abend code values, REQUEST=AUTH uses these codes.)
4	Try the RACROUTE REQUEST=AUTH call again; invoke the RACROUTE REQUEST=AUTH preprocessing exit routine. (Any values in the return or abend code fields are ignored and the fields are reset to zero. Other fields are not affected. In particular, the INSTLN value is not reinitialized; this preserves any information placed in it by the preprocessing or postprocessing exit routine.)

Note: If register 15 contains any other value, RACROUTE REQUEST=AUTH issues an abend code (382) that indicates a non-valid exit return code.

Possible uses of the exits

Allowing access when RACF is inactive

When RACF is inactive, any attempt to access a protected resource is passed to the RACROUTE REQUEST=AUTH preprocessing exit. The exit can determine whether or not to allow the access to proceed. If, for example, you want to allow one or more specific users to perform recovery operations, the exit must select those users. (If RACF is inactive, the normal privileges of OPERATIONS cannot be used, because the RACF database might not be available to verify that a user is so authorized.) You should consider whether user IDs or batch job names authorized by the exits when RACF is inactive should also be allowed to use the system if RACF is running normally.

If the RACROUTE REQUEST=AUTH preprocessing exit routine neither grants nor denies access to the data set, RACF failsoft processing can prompt the operator.

Protecting the user's resources from the user

Users can accidentally delete their own data. Suppose, for example, that a user wishes to delete a library member, but forgets to include the member name in the command. The user issues:

```
DELETE  USER.LIB
```

instead of

```
DELETE  USER.LIB(progname)
```

With ALTER authority to USER.LIB, the result is the loss of the entire library.

To delete a member requires UPDATE authority, whereas deletion of the whole library requires ALTER. By default, the creator of a data set—a library is only a special use of a data set—automatically has ALTER authority to it. Therefore, the user is susceptible to this exposure.

For group data sets (libraries), creators are explicitly in the access list and can therefore take positive steps to reduce their own authority to UPDATE. They are

then unable to delete the group data set accidentally. When they do want to delete it, then, still being the owner of the data set, they can restore ALTER authority by using the PERMIT command.

While this is very simple for group data sets, user data sets do not have their creators in the access list. Creators have ALTER authority by virtue of the naming convention; the high-level qualifier and user ID match. There are, therefore, no user IDs for the creators to remove, and their data is still vulnerable to their own errors.

To provide the same facility for user data sets as for group data sets, you must use the RACF exits. In the RACROUTE REQUEST=AUTH exit, you can determine whether the user is seeking to scratch one of his or her own data sets. Under these circumstances, the exit can invalidate access with the naming convention, by blanking out the QUALIFIER field and allowing access only as specified in the access list. When users really want to delete entire data sets, users can authorize themselves explicitly because they are the owners.

Controlling access of shared user IDs

The certificate mapping profile maps an issuers distinguished user name to an Internet user ID. The certificate mapping profiles maps many certificates to the same user ID. A certificate that fits the mapping profile receives full use of that user ID, meaning that the user has the same rights and privileges as the user ID being used.

In some cases, this might not be the correct thing to do. For example,

- The shared user ID might need access to a resource that is not normally granted to the ID but is normally accessed by the user who is using the ID.
- The shared user ID might have access to a resource that is not not normally granted to the individual user who is using the shared ID, in which case the access should be denied.

Using the RACROUTE REQUEST=AUTH preprocessing exit (ICHRCX01), you can check the X500 name (ACEEX5PR) to determine which accesses and privileges the user should have. The X500 name uniquely identifies the user of a shared user ID.

To override the privileges normally granted to the shared user ID, you need to write a preprocessing exit.

1. The exit checks the contents of the X500 name and the user ID.
2. The X500 name (ACEEX5PR) points to a control block containing the issuer's and the subject's distinguished name.
3. The exit compares the contents and permits or denies privileges to resources based on the privileges of the specific user of the shared user ID.

RACROUTE REQUEST=DEFINE exits

The purpose of RACROUTE REQUEST=DEFINE is to define, modify, and delete discrete or generic DASD data set profiles, or profiles for any resource defined by classes in the class descriptor table, or to determine the user's authority to create, delete, or rename a resource protected by a profile.

You can use the RACROUTE REQUEST=DEFINE exits to cause all allocation authorization checks (DEFINE requests from DADSM ALLOCATE) to be accepted, regardless of the user's authority.

Many of the values passed to the RACROUTE REQUEST=DEFINE preprocessing and postprocessing exits are derived from the parameters specified on the RACROUTE REQUEST=DEFINE macro. For details on this macro, see *z/OS Security Server RACROUTE Macro Reference*. If you want an exit to modify parameters that were specified on the original RACROUTE REQUEST=DEFINE request, the recommended method is to do this from the preprocessing exit, ICHRDX01. If you make changes to these parameters from the postprocessing exit, ICHRDX02, RACF might not recognize the changes.

The RACROUTE REQUEST=DEFINE exits must not do anything to prevent the creation of profiles in the DIGTCERT class.

Extended addressing

In many cases RACF must copy caller-supplied parameter areas to an area below 16MB, so that the exit can address the parameter areas. However, when the RACROUTE REQUEST=DEFINE exit routines receive the ACCLVL parameter or INSTLN parameters, RACF does not know the format or length of the parameters being passed, and therefore cannot copy the parameters into 24-bit storage. Because the parameter list pointing to these parameters is in 24-bit storage, you must modify the exit routines to handle the parameters if the exit routines access these areas, and if they are passed by callers in 31-bit mode.

Automatic direction of application updates

When the RACROUTE REQUEST=DEFINE exit routines receive the ACCLVL or the INSTLN parameters, RACF does not know the format of the parameters. As a result, if automatic direction of application updates is active, RACF does not know what information to propagate. If you want RACF to propagate these parameters, you must have a RACROUTE REQUEST=DEFINE exit, and your exit must specify what information RACF is to propagate. An exit does this by setting fields in the RDXP parameter list. If an exit does not set these fields, by default RACF does not propagate the parameters. For information on these fields, see *z/OS Security Server RACF Data Areas*. An exit might also need to recognize when these parameters have already been processed by the exit on another system, and take appropriate action. The RDDFPROP flag in the RACROUTE REQUEST=DEFINE parameter list, RDDFL, indicates whether the DEFINE request was directed from another system.

Preprocessing exit (ICHRDX01)

The RACROUTE REQUEST=DEFINE preprocessing exit routine must be named ICHRDX01. It is entered before the definition, modification, or deletion of resource profiles.

The exit must be reentrant and is invoked in supervisor state, with protection key 0, with no locks held.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

When the RACROUTE REQUEST=DEFINE preprocessing exit receives control for the DATASET class, RACF has already processed the naming convention table, if there is one.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=DEFINE exit parameter list, RDXP.

Return codes from the RACROUTE REQUEST=DEFINE preprocessing exit

When the RACROUTE REQUEST=DEFINE preprocessing exit routine returns control, register 15 should contain one of the following return codes. Do not confuse these return codes with the return codes from the RACROUTE REQUEST=DEFINE macro, which are documented in *z/OS Security Server RACROUTE Macro Reference*.

Hex	(Decimal)	Meaning
0	(0)	Exit-routine processing is complete. Normal processing is to continue.
4	(4)	The request is not accepted and is to be failed.
8	(8)	The request is accepted. No more processing is to be performed.
C	(12)	The request is accepted. Processing continues, but authorization checking is bypassed.

Notes:

1. If register 15 contains any other value other than those listed above, RACROUTE REQUEST=DEFINE issues a completion code (385) that indicates a non-valid exit return code.
2. A return code of 4 from the preprocessing exit for an ADDVOL request results in abend 385-4 (non-valid return code).
3. The RACROUTE REQUEST=DEFINE exit parameter list points to the naming-convention parameter list. For a description of what happens if you change the naming-convention parameter list when you code the RACROUTE REQUEST=DEFINE preprocessing exit, see the description of the naming-convention exit parameter list, CNXP, in *z/OS Security Server RACF Data Areas*.

Postprocessing exit (ICHRDX02)

The RACROUTE REQUEST=DEFINE postprocessing exit routine must be named ICHRDX02. It is entered after authorization checking and profile retrieval but before a new profile is created or before any changes are made to the RACF database.

The exit must be reentrant and is invoked in supervisor state, with protection key 0, with no locks held.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=DEFINE exit parameter list, RDXP.

Return codes from the RACROUTE REQUEST=DEFINE postprocessing exit

When the RACROUTE REQUEST=DEFINE postprocessing exit routine returns control, register 15 should contain one of the following return codes. Do not confuse these return codes with the return codes from the RACROUTE REQUEST=DEFINE macro, which are documented in *z/OS Security Server RACROUTE Macro Reference*.

Code	Meaning
0	Exit-routine processing is complete. Normal processing is to continue.
4	Retry the REQUEST=DEFINE function; invoke the RACROUTE

REQUEST=DEFINE preprocessing routine. (Before a retry, the return code, reason code, completion code, access authority, owner, level, and auditing fields are reset to zeros.)

Note: If register 15 contains any other value, RACROUTE REQUEST=DEFINE issues a completion code (385) that indicates a non-valid exit return code.

RACROUTE REQUEST=FASTAUTH exits

RACROUTE REQUEST=FASTAUTH examines the auditing and global options in effect for the resource while determining the access authority of the caller. The FASTAUTH request returns a reason code that indicates whether the access attempt should be logged. The RACROUTE REQUEST=FASTAUTH exits allow the installation to make additional security checks or to instruct RACROUTE REQUEST=FASTAUTH to either accept or fail a request.

Notes:

1. The RACROUTE REQUEST=FASTAUTH exits do not get control during authorization requests for the PROGRAM class and cannot be used to affect PROGRAM processing.
2. When the FASTAUTH request is invoked for the UNIXPRIV class, the FASTAUTH service is called directly from a callable service, and the SAF router exit, ICHRTX00, is not called.

Preprocessing exits (ICHRFX01 and ICHRFX03)

There are two RACROUTE REQUEST=FASTAUTH preprocessing exits. ICHRFX01 is used for non-cross-memory calls. ICHRFX03 is used for cross-memory calls.

Exception: When a FASTAUTH request is invoked for the UNIXPRIV class, ICHRFX03, if present, is always called instead of ICHRFX01, even in non-cross-memory mode.

The preprocessing exits are entered before the RACROUTE REQUEST=FASTAUTH service routine performs authorization checking.

ICHRFX01

This exit must be reentrant. The exit must have RMODE(24) and AMODE(ANY).

It is extremely important that the writer of the RACROUTE REQUEST=FASTAUTH exit routine be aware of the environment in which the routine will be executing. This routine is *not* invoked using standard linkage conventions. Its running environment offers limited function as indicated in the following list:

1. The execution key is unpredictable.
2. The exit might receive control in either supervisor or problem state.
3. The exit might or might not be given control APF-authorized.
4. The exit might be given control in SRB mode; that is, the REQUEST=FASTAUTH might have been issued by a caller running as an SRB.
5. The exit should not issue any SVCs.
6. The exit routine might be given control in either 24- or 31-bit mode.
7. The exit is responsible for saving and restoring certain registers it uses. The RACROUTE REQUEST=FASTAUTH exit parameter list contains a pointer (RFXWA) to a 16-word work area. The exit can use the first 15 words of this area to save and restore registers.

   On entry to the exit:

- R1 contains the address of the exit parameter list, which contains a pointer to the 16-word FASTAUTH work area. In the FASTAUTH work area:
  - The 12th word contains the RACF reason code that REQUEST=FASTAUTH processing has determined up to this point.
  - The 13th word contains the RACF return code that REQUEST=FASTAUTH processing has determined up to this point.
  - The 14th word contains 0.
  - The 15th word contains 0, and can be used by the exit to pass information to the postprocessing exit or the FASTAUTH caller.
- R14 contains the return address.
- R15 contains the address of the exit entry point.

If the exit changes register 5, the exit must save that register and restore it before returning to RACF. The exit can modify any of the other registers without restoring the value the register had on entry to the exit.

Of course, the R14 value is needed to return to RACF.

8. If ICHRFC00 or IGC0013{ is placed in the fixed link pack area (FLPA), the exit should also be in FLPA.

The RACROUTE REQUEST=FASTAUTH ICHRFX01 parameter list is the RACROUTE REQUEST=FASTAUTH input parameter list. Either the RFXP or the FAST mapping in *z/OS Security Server RACF Data Areas* maps the parameter list.

Return codes from the ICHRFX01 preprocessing exit: On return from the exit routine, RACROUTE REQUEST=FASTAUTH checks register 15 for one of the following codes:

Code	Meaning
0	RACROUTE REQUEST=FASTAUTH is to continue processing the request.
4	RACROUTE REQUEST=FASTAUTH is to fail the request. RACROUTE REQUEST=FASTAUTH will return to its caller with a SAF return code of 8 and a RACF return code (in SAFPRRET) of 8.
8	RACROUTE REQUEST=FASTAUTH is to accept the request. RACROUTE REQUEST=FASTAUTH performs no further authorization processing, and returns control to its caller with a SAF return code of 0 and a RACF return code (in SAFPRRET) of 0.

Any other code from the exit is treated as an error, and RACROUTE REQUEST=FASTAUTH returns to its caller with a SAF return code of 8 and a RACF return code (in SAFPRRET) of X'10'.

If the exit returns with R15=4 or 8, the exit is responsible for setting the 12th word of the work area (FASTAUTH reason code) that RFXWA points to, as follows:
- 0 if the exit does not want the FASTAUTH issuer to cause auditing of this request
- 4 if the exit does want the FASTAUTH issuer to cause auditing of the request

After FASTAUTH returns to the caller, the 14th word of the work area and R1 point to a profile if all of the following are true:
- The profiles were not in a data space,
- The SAF return code is 0 or 8, and
- A profile, rather than the preprocessing exit, was used to make the decision.

Note: If the preprocessing exit returned to RACF with return code 4 or 8, no profile address is returned to the caller.

The 15th word of the work area can be used to communicate between the preprocessing exit and the postprocessing exit, if any. It can also be used to communicate between the exits and RACF's caller.

ICHRFX03

This exit must be reentrant.

The exit can have any RMODE, and must have AMODE(31) or AMODE(ANY). It is always invoked in AMODE(31).

The exit is invoked in primary ASC mode.

This exit is passed the parameter list FXAP, which is located in the primary address space. The parameter list contains the address of the ICHRFX01 parameter list (mapped by RFXP or FAST), which is actually the parameter list in the caller's storage and under the caller's key with which FASTAUTH was invoked. The parameter list in turn points to the 16-word FASTAUTH work area.

It is extremely important that the writer of the RACROUTE REQUEST=FASTAUTH exit routine be aware of the environment in which the routine will be executing. This routine is *not* invoked using standard linkage conventions. Its running environment offers limited function as indicated in the following list:

1. The exit is invoked in supervisor state, with protection key 0, with no locks held. Writers of this exit who are concerned about integrity might want to consider having any reference or setting of fields in the ICHRFX01 parameter list or 16-word work area done under the caller's key. The caller's key can be obtained by issuing the ESTA instruction.
2. The exit must not issue any SVCs.
3. The exit routine always receives control in 31-bit mode.
4. The exit is responsible for saving and restoring certain registers it uses. The ICHRFX01 parameter list (RFXP) contains a pointer (RFXWA) to a 16-word work area. The exit can use the first 15 words of this area to save and restore registers.

   On entry to the exit:

   - R1 contains the address of the exit parameter list (FXAP) which contains the address of the ICHRFX01 parameter list (RFXP or FAST) which contains a pointer to the work area. In the work area:
     - The 12th word contains the RACF reason code that REQUEST=FASTAUTH processing has determined up to this point.
     - The 13th word contains the RACF return code that REQUEST=FASTAUTH processing has determined up to this point.
     - The 14th word contains 0.
     - The 15th word contains 0, and can be used by the exit to pass information to the postprocessing exit or the FASTAUTH caller.
   - R14 contains the return address.
   - R15 contains the address of the exit entry point.

   If the exit changes register 12, the exit must save and restore it before returning to RACF. The exit can modify any other register without restoring the value the register had on entry to the exit.

Of course the R14 value is needed to return to RACF.

5. If IGC0013{ is placed in the fixed link pack area (FLPA), the exit should also be in the fixed link pack area (FLPA).

The RACROUTE REQUEST=FASTAUTH ICHRFX03 parameter list is mapped by FXAP (see *z/OS Security Server RACF Data Areas*). It points to the ICHRFX01 parameter list, which is mapped by RFXP or FAST.

When the ACEELET keyword is specified on the RACROUTE REQUEST=FASTAUTH macro, you must access the ACEE using the ALET in the RFXALET field of the RFXP parameter list. Otherwise, you can access the ACEE in the current HOME memory space. For cross-memory callers, the ACEE must be accessed using an ALET of 2.

When the RACROUTE REQUEST=FASTAUTH macro specifies the ENVRIN keyword, the RFXPENVR field in the parameter list points to an ENVR object, and the ACEE address in the parameter list points to a *temporary* ACEE, built only for FASTAUTH processing. The exit can expect the RFXPENVR field to be present only if the RFXPVERS version indicator has a value of 2 or higher. This temporary ACEE is built in FASTAUTH's storage, which is obtained in key 0, and might not be in the subpool indicated by the ACEE in the ACEESP field. FASTAUTH installation exits can remain in the key in which they are called when the ENVRIN keyword is present, because this keyword can only be specified by callers running in supervisor state or system key.The exit should not obtain storage and anchor it in the temporary ACEE. Installation data pointed to by ACEEIEP in the original ACEE is only present in the temporary ACEE if it is in standard format. If the installation data is not in standard format (indicated by IRRACX01 or IRRACX02 returning a range table at compression time), RACF sets ACEEIEP to 0 in the temporary ACEE. If ACEEIEP does point to standard data, the subpool specification might not be accurate. The exit should not change the data pointed to by ACEEIEP, and must not delete it.

The exit must be aware that the temporary ACEE might be created from an ENVR object that originated on another system. If the ACEE was created from an ENVR object that originated on another system, the ACEEXNVR bit is set. If the FASTAUTH exits need to know the exact origin of the ACEE information, you can use the ACEEIEP installation data field. An exit on the remote system (for example, the RACROUTE REQUEST=VERIFY(X) postprocessing exit, ICHRIX02) would need to update the installation data field when the ACEE is created.

Return codes from the ICHRFX03 preprocessing exit: On return from the exit routine, RACROUTE REQUEST=FASTAUTH checks register 15 for one of the following codes:

Code	Meaning
0	RACROUTE REQUEST=FASTAUTH is to continue processing the request.
4	RACROUTE REQUEST=FASTAUTH is to fail the request. RACROUTE REQUEST=FASTAUTH returns to its caller with a SAF return code of 8 and a RACF return code (in SAFPRRET) of 8.
8	RACROUTE REQUEST=FASTAUTH is to accept the request. RACROUTE REQUEST=FASTAUTH performs no further authorization processing, and returns control to its caller with a SAF return code of 0 and a RACF return code (in SAFPRRET) of 0.

Any other code from the exit is treated as an error, and RACROUTE REQUEST=FASTAUTH returns to its caller with a SAF return code of 8 and a RACF return code (in SAFPRRET) of X'10'.

If the exit returns with R15=4 or 8, the exit is responsible for setting the 12th word of the work area (FASTAUTH reason code) that RFXWA points to, as follows:
- 0 if the exit does not want the FASTAUTH issuer to cause auditing of this request
- 4 if the exit does want the FASTAUTH issuer to cause auditing of the request.

After FASTAUTH returns to the caller, the 14th word of the work area and R1 point to a profile if all of the following are true:
- The profiles were not in a data space,
- The SAF return code is 0 or 8, and
- A profile, rather than the preprocessing exit, was used to make the decision.

Note: If the preprocessing exit returned to RACF with return code 4 or 8, no profile address is returned to the caller.

The 15th word of the work area can be used to communicate between the preprocessing exit and the postprocessing exit, if any. It can also be used to communicate between the exits and RACF's caller.

Postprocessing exits (ICHRFX02 and ICHRFX04)

There are two RACROUTE REQUEST=FASTAUTH postprocessing exits: ICHRFX02 and ICHRFX04. The following table summarizes the conditions under which each postprocessing exit is called:

If these conditions are met:	Then these postprocessing exits are called:
ICHRFX01 or ICHRFX03 set a nonzero return code	neither postprocessing exit is called
FASTAUTH is issued in cross-memory mode	ICHRFX04
FASTAUTH is processing the UNIXPRIV class	ICHRFX04
FASTAUTH is processing a class RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES	ICHRFX04, then ICHRFX02
FASTAUTH is processing a class RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=NO	ICHRFX02
FASTAUTH is processing a class RACLISTed by SETROPTS RACLIST	ICHRFX04, then ICHRFX02
FASTAUTH is issued in a non-cross-memory environment and no other conditions above are satisfied.	ICHRFX02

The sequence of pre- and post- processing exit invocation, FASTAUTH authorization processing, and auditing (when FASTAUTH performs auditing due to LOG=ASIS or LOG=NOFAIL), is:

Note: The following table applies to all classes specified on the FASTAUTH request except the UNIXPRIV class. For that class, the sequence shown in the third column for cross-memory mode applies; only the ICHRFX03 and ICHRFX04 exits are invoked.

Non-cross-memory GLOBAL=NO	Non-cross-memory for both GLOBAL=YES and SETROPTS RACLIST	Cross-memory or ACEEALET=*alet-addr* or ENVIR=*envr-data-addr* regardless of how the class is RACLISTed
1. ICHRFX01	1. ICHRFX01	1. ICHRFX03
2. Auth processing	2. Auth processing	2. Auth processing
3. ICHRFX02	3. ICHRFX04	3. ICHRFX04
4. Auditing	4. Auditing	4. Auditing
	5. ICHRFX02	

Note that in the case where FASTAUTH is not running in cross-memory mode and is processing profiles RACLISTed to a data space with GLOBAL=YES, RACF performs logging based on the return and reason code set by:

- ICHRFX01 or ICHRFX04, if they exist
- If ICHRFX01 and ICHRFX04 do not exist, by one of the following::
  - The default return code defined for the class in the class descriptor table (CDT)
  - FASTAUTH processing done before ICHRFX02 gets control

Any return and reason code set by ICHRFX02 in this case is not reflected in the auditing done by FASTAUTH, but is processed as described in "ICHRFX02."

Default return code processing occurs prior to auditing. If the profile was not found and the postprocessing exit did not change the return code, FASTAUTH uses the default return code from the class descriptor table (CDT). The default return code, if used, is reflected in the auditing done by FASTAUTH.

ICHRFX02

This exit must be reentrant.

The exit must have RMODE(24) and AMODE(ANY).

It is extremely important that the writer of the RACROUTE REQUEST=FASTAUTH exit routine be aware of the environment in which the routine will be executing. This routine is *not* invoked using standard linkage conventions. Its running environment offers limited function as indicated in the following list:

1. The execution key is unpredictable.
2. The exit might receive control in either supervisor or problem state.
3. The exit might or might not be given control APF-authorized.
4. The exit might be given control in SRB mode; that is, the REQUEST=FASTAUTH might have been issued by a caller running as an SRB.
5. The exit should not issue any SVCs.
6. The exit routine might be given control in either 24- or 31-bit mode.
7. The exit is responsible for saving and restoring certain registers it uses. The RACROUTE REQUEST=FASTAUTH exit parameter list contains a pointer (RFXWA) to a 16-word work area. The exit can use the first 15 words of this area to save and restore registers.

   On entry to the exit:
   - R1 contains the address of the exit parameter list, which contains a pointer to the 16-word FASTAUTH work area. In the FASTAUTH work area:

  - The 12th word contains the RACF reason code that REQUEST=FASTAUTH processing has determined up to this point.
  - The 13th word contains the RACF return code that REQUEST=FASTAUTH processing has determined up to this point.
  - The 14th word contains 0 if no profile protecting the resource was found or if the class was RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES or by SETROPTS RACLIST. Otherwise it contains a pointer to the profile. The profile is mapped by RACRPE within the ISP data area, documented in *z/OS Security Server RACF Data Areas*.
  - The 15th word contains 0 or information set by the ICHRFX01 or ICHRFX04 exits if they were invoked and set this word to a value.
- R14 contains the return address.
- R15 contains the address of the exit entry point.

If the exit changes register 5, the exit must save that register and restore it before returning to RACF. The exit can modify any of the other registers without restoring the value the register had on entry to the exit.

Of course, the R14 value is needed to return to RACF.

8. If the RACROUTE REQUEST=FASTAUTH routine (ICHRFC00 or IGC0013{) is placed in the fixed link pack area (FLPA), the exit should also be in the fixed link pack area (FLPA).

The RACROUTE REQUEST=FASTAUTH ICHRFX02 parameter list is the RACROUTE REQUEST=FASTAUTH input parameter list. Either the RFXP or the FAST mapping in *z/OS Security Server RACF Data Areas* maps the parameter list.

Return codes from the ICHRFX02 postprocessing exit: The postprocessing exit routine must return to the RACROUTE REQUEST=FASTAUTH service routine with a return code of 0. RACROUTE REQUEST=FASTAUTH treats any other return code as an error and returns to the RACROUTE issuer with a SAF return code of 8 and a RACF return code (in SAFPRRET) of X'10'.

In some cases, the RACF return code passed to the exit is changed by RACF before it is returned to the caller of RACROUTE REQUEST=FASTAUTH. If the return code passed to the exit is 4, but the default return code for the class is not 4 because a profile was not found, the final RACF return code is the default return code for the class.

When ICHRFX02 returns to RACROUTE REQUEST=FASTAUTH processing, the 16-word work area pointed to by the parameter list should contain the following information:

- The 12th word contains the RACF reason code that REQUEST=FASTAUTH passes back to its caller in SAFPRREA. Exception: If word 13 contains a 4 or a value greater than 8 when the exit returns, then REQUEST=FASTAUTH sets SAFPRREA to 0.

  Note: If the class being processed was RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES or by SETROPTS RACLIST, and auditing was to be performed by FASTAUTH (via LOG=ASIS or LOG=NOFAIL), FASTAUTH has already done the auditing based on its authorization processing or return and reason codes set by ICHRFX01 or ICHRFX04. Setting the reason code to indicate 'log' now might result (depending on the return code value) in that reason code being returned to FASTAUTH's

caller, but does not cause FASTAUTH to perform auditing. See "Postprocessing exits (ICHRFX02 and ICHRFX04)" on page 290 for a discussion of the sequence in which the exits are invoked and auditing is performed.

- The 13th word contains the RACF return code that REQUEST=FASTAUTH passes back to its caller in SAFPRRET. If this value is 0, the SAF return code will also be 0. If this value is a 4, X'0C', X'1C', or X'20', the SAF return code will be 4. If this value is anything else, the SAF return code will be 8.
- The 14th word contains either 0 or the pointer to the profile passed to the exit on entry. (If auditing is yet to be performed by FASTAUTH, audit information is taken from the profile addressed by this field.)
- The 15th word contains either 0 or a value to be passed to the FASTAUTH caller as set by a previously invoked pre- or post- processing exit, or by ICHRFX02 itself.

ICHRFX04

This exit must be reentrant.

The exit can have any RMODE, and must have AMODE(31) or AMODE(ANY). It is always invoked in AMODE(31).

The exit is invoked in primary ASC mode.

This exit is passed the parameter list FXAP, which is located in the primary address space. The parameter list contains the ALET to the data space (GLOBAL=YES or SETROPTS RACLIST) or address space (cross-memory GLOBAL=NO) and the pointer to the profile used for authority checking. The profile is mapped by RACRPE within the ISP data area, documented in *z/OS Security Server RACF Data Areas*. FXAP also contains the address of the ICHRFX02 parameter list (RFXP or FAST), which is actually the parameter list in the caller's storage and under the caller's key with which FASTAUTH was invoked. The parameter list in turn points to the 16-word FASTAUTH work area.

It is extremely important that the writer of the RACROUTE REQUEST=FASTAUTH exit routine be aware of the environment in which the routine will be executing. This routine is *not* invoked using standard linkage conventions. Its running environment offers limited function as indicated in the following list:

1. The exit is invoked in supervisor state, with protection key 0, with no locks held. Writers of this exit who are concerned about integrity might want to consider having any reference or setting of fields in the ICHRFX02 parameter list or 16-word work area done under the caller's key. The caller's key can be obtained by issuing the ESTA instruction.
2. The exit must not issue any SVCs.
3. The exit routine always receives control in 31-bit mode.
4. The exit is responsible for saving and restoring certain registers it uses. The ICHRFX02 parameter list (RFXP) contains a pointer (RFXWA) to a 16-word work area. The exit can use the first 15 words of this area to save and restore registers.

   On entry to the exit:

   - R1 contains the address of the exit parameter list (FXAP) which contains the address of the ICHRFX02 parameter list (RFXP) which contains a pointer to the 16-word FASTAUTH work area. In the FASTAUTH work area:
     - The 12th word contains the RACF reason code that REQUEST=FASTAUTH processing has determined up to this point.

  - The 13th word contains the RACF return code that REQUEST=FASTAUTH processing has determined up to this point.
  - The 14th word contains 0.
  - The 15th word contains 0 or information set by the ICHRFX01 or ICHRFX03 exits if they were invoked and set this word to a value.
- R14 contains the return address.
- R15 contains the address of the exit entry point.

If the exit changes register 12, the exit must save and restore it before returning to RACF. The exit can modify any other register without restoring the value the register had on entry to the exit.

Of course the R14 value is needed to return to RACF.

5. If the RACROUTE REQUEST=FASTAUTH routine (ICHRTC00 or IGC0013{) is placed in the fixed link pack area (FLPA), the exit should also be in the fixed link pack area (FLPA).

The RACROUTE REQUEST=FASTAUTH ICHRFX04 parameter list is mapped by FXAP. See *z/OS Security Server RACF Data Areas*. The ICHRFX04 parameter list points to the ICHRFX02 parameter list, which is mapped by RFXP or FAST.

When the ACEELET keyword is specified on the RACROUTE REQUEST=FASTAUTH macro, you must access the ACEE using the ALET in the RFXALET field of the RFXP parameter list. Otherwise, you can access the ACEE in the current HOME memory space. For cross-memory callers, the ACEE must be accessed using an ALET of 2.

When the RACROUTE REQUEST=FASTAUTH macro specifies the ENVRIN keyword, the RFXPENVR field in the parameter list points to an ENVR object, and the ACEE address in the parameter list points to a *temporary* ACEE, built only for FASTAUTH processing. The exit can expect the RFXPENVR field to be present only if the RFXPVERS version indicator has a value of 2 or higher. This temporary ACEE is built in FASTAUTH's storage, which is obtained in key 0, and might not be in the subpool indicated by the ACEE in the ACEESP field. FASTAUTH installation exits can remain in the key in which they are called when the ENVRIN keyword is present, because this keyword can only be specified by callers running in supervisor state or system key. The exit should not obtain storage and anchor it in the temporary ACEE. Installation data pointed to by ACEEIEP in the original ACEE is only present in the temporary ACEE if it is in standard format. If the installation data is not in standard format (indicated by IRRACX01 or IRRACX02 returning a range table at compression time), RACF sets ACEEIEP to 0 in the temporary ACEE. If ACEEIEP does point to standard data, the subpool specification might not be accurate. The exit should not change the data pointed to by ACEEIEP, and must not delete it.

The exit must be aware that the temporary ACEE might be created from an ENVR object that originated on another system. If the ACEE was created from an ENVR object that originated on another system, the ACEEXNVR bit is set. If the FASTAUTH exits need to know the exact origin of the ACEE information, you can use the ACEEIEP installation data field. An exit on the remote system (for example, the RACROUTE REQUEST=VERIFY(X) postprocessing exit, ICHRIX02) would need to update the installation data field when the ACEE is created.

Return codes from the ICHRFX04 postprocessing exit: The postprocessing exit routine must return to the RACROUTE REQUEST=FASTAUTH service routine with

a return code of 0. RACROUTE REQUEST=FASTAUTH treats any other return code as an error, and returns to the RACROUTE issuer with a SAF return code of 8 and a RACF return code (in SAFPRRET) of X'10'.

In some cases, the RACF return code passed to the exit is changed by RACF before it is returned to the caller of RACROUTE REQUEST=FASTAUTH. If the return code passed to the exit is 4, but the default return code for the class is not 4 because a profile was not found, the final RACF return code is the default return code for the class.

When ICHRFX04 returns to RACROUTE REQUEST=FASTAUTH processing, the 16-word work area pointed to by the parameter list should contain the following information:

- The 12th word contains the RACF reason code that REQUEST=FASTAUTH passes back to its caller in SAFPRREA. Exception: If the 13th word contains a 4 or a value greater than 8 when the exit returns, then REQUEST=FASTAUTH sets SAFPRREA to 0.
- The 13th word contains the RACF return code that REQUEST=FASTAUTH passes back to its caller in SAFPRRET. If this value is 0, the SAF return code will also be 0. If this value is a 4, X'0C', X'1C', or X'20', the SAF return code will be 4. If this value is anything else, the SAF return code will be 8.
- The 14th word always contains a zero.
- The 15th word contains 0 or a value to be passed to the ICHRFX02 exit or to the FASTAUTH caller.

Possible uses of the exits

Controlling access of shared user IDs

The certificate mapping profile maps an issuer's distinguished user name to an Internet user ID. The certificate mapping profiles maps many certificates to the same user ID. A certificate that fits the mapping profile receives full use of that user ID, meaning that the user has the same rights and privileges as the user ID being used.

In some cases, this might not be the correct thing to do. For example,

- The shared user ID might need access to a resource that is not normally granted to the ID but is normally accessed by the user who is using the ID.
- The shared user ID might have access to a resource that is not not normally granted to the individual user who is using the shared ID, in which case the access should be denied.

Using the RACROUTE REQUEST=FASTAUTH preprocessing exits (ICHRFX01 and ICHRFX03), you can check the X500 name (ACEEX5PR) to determine which accesses and privileges the user should have. The X500 name uniquely identifies the user of the shared user ID.

To override the privileges normally granted to the shared user ID, you need to write a preprocessing exit.

1. The exit checks the contents of the X500 name and the user ID.
2. The X500 name (ACEEX5PR) points to a control block containing the issuer's and the subject's distinguished name.
3. The exit compares the contents and permits or denies privileges to resources based on the privileges of the specific user of the shared user ID.

RACROUTE REQUEST=LIST exits

RACROUTE REQUEST=LIST is used to build in-storage (resident) copies of general-resource profiles. Both the RACROUTE REQUEST=AUTH and RACROUTE REQUEST=FASTAUTH routines can use these resident profiles for authorization checking. The RACROUTE REQUEST=LIST pre- and postprocessing exit (ICHRLX01) and the selection exit (ICHRLX02) allow the installation to modify REQUEST=LIST processing options and to resolve conflicts between new and existing profile information.

RACROUTE REQUEST=LIST processing is as follows:

1. RACF calls the preprocessing exit routine to perform initialization of the installation environment.
2. If the resource class being processed has a resource-group class associated with it, then for every entity in the resource group class:
   a. REQUEST=LIST individually processes each member in the resource-group entity.
   b. REQUEST=LIST calls the selection exit routine to resolve conflicts between the information associated with the member resource currently being processed and a previously-built profile for that member, if, for example, a resource is a member of more than one grouping entity.
   c. REQUEST=LIST builds an in-storage profile for the member resource (or updates the previously-built profile).
3. For each resource in the class (or specified by the LIST option):
   a. REQUEST=LIST calls the selection exit routine to resolve conflicts between the information associated with the resource currently being processed and a previously-built profile for that resource, if, for example, a resource has an individual profile in a RACF data set and is a member of one or more resource-group entities.)
   b. REQUEST=LIST builds an in-storage profile for the resource (or updates the previously-built profile).
4. RACF calls the postprocessing exit routine to clean up the installation environment.

RACROUTE REQUEST=LIST is used by products requiring high-performance authorization checking (such as IMS and CICS). They then use the RACROUTE REQUEST=FASTAUTH service, possibly followed by the RACROUTE REQUEST=AUTH service, to do authorization checking. If you need to create an authorization checking exit for IMS or CICS, you may need to use a FASTAUTH exit or both a FASTAUTH exit and an AUTH exit.

ICHRLX01 is entered before RACROUTE REQUEST=LIST builds any in-storage profiles of RACF-defined resources and again after the profiles have been built (at the end of REQUEST=LIST processing). ICHRLX02 is entered as each profile is being built.

A resource name can appear in more than one resource-group profile and at the same time can have a profile of its own. RACROUTE REQUEST=LIST resolves conflicts between these multiple profiles for the following fields:

- UACC
- LEVEL
- Audit options
- Global audit options
- Installation data

- Access list entries
- Owner
- Categories
- SECLABEL

The RACROUTE REQUEST=LIST preprocessing exit can specify general rules for this resolution, such as to use the most or the least restrictive option, or to use the first or the last value found. The RACROUTE REQUEST=LIST selection exit (which is passed the profile built to that point and the new values to be resolved) can make specific decisions. ICHRLX02 is entered as each profile is being built. The RACROUTE REQUEST=LIST selection exit can also resolve conflicts for the OWNER field.

If there are no exits to invoke, RACF checks all the profiles and does the following:
- Uses the most restrictive UACC
- For any particular user, uses the least restrictive of the access entries
- Uses the highest security level
- Does auditing if requested by any of the profiles
- Combines category lists
- Chooses the first SECLABEL field found

Pre- and postprocessing exit (ICHRLX01)

The RACROUTE REQUEST=LIST pre- and postprocessing exit must be named ICHRLX01.

This exit must be reentrant and is invoked in supervisor state, under protection key 0, with no locks held.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=LIST exit parameter list, RLX1P.

Return codes from ICHRLX01

On return from the ICHRLX01 exit routine, RACROUTE REQUEST=LIST checks register 15 for one of the following return codes:

Code	Meaning
0	The LIST request is to continue processing.
4	The LIST request is to terminate processing. For a return code, RACROUTE REQUEST=LIST uses the return code passed as a parameter and possibly modified by the exit. A code of 0 returned after a call for postprocessing is treated the same way as code 4.

Any other return code is treated as an error, and RACROUTE REQUEST=LIST returns to its caller with a return code of 14 (hexadecimal).

Selection exit (ICHRLX02)

The RACROUTE REQUEST=LIST selection exit must be named ICHRLX02.

This exit must be reentrant and is invoked in supervisor state, under protection key 0, with no locks held.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=LIST exit parameter list, RLX2P.

Return codes from the RACROUTE REQUEST=LIST selection exit

On return from the RACROUTE REQUEST=LIST selection exit routine, REQUEST=LIST checks register 15 for one of the following return codes:

Hex	(Decimal)	Meaning
0	(0)	REQUEST=LIST is to continue processing.
4	(4)	REQUEST=LIST is not to merge access lists. The working copy of the profile is unchanged. Note that the exit can modify the effect of this return code by modifying the working-profile access list.
8	(8)	REQUEST=LIST is to mark the resource as being logically undefined; this makes the resource name unavailable within the in-storage profile structure. In particular, if the name is encountered again, it will be processed as if it were the first occurrence.
C	(12)	REQUEST=LIST is to terminate all processing. The return code passed to the exit in the preprocessing exit's list will be used as the LIST request's return code. The exit can set the return-code parameter to whatever value it desires. Its initial value (0) is used unless the exit explicitly modifies it.

Any other return code is treated as an error, and RACROUTE REQUEST=LIST returns to its caller with a return code of 14 (hexadecimal).

RACROUTE REQUEST=VERIFY(X) exits

A RACROUTE REQUEST=VERIFY or RACROUTE REQUEST=VERIFYX request is used to determine whether a user ID is defined to RACF and whether the user has supplied a valid password and group name. During TSO logon processing, the VERIFY request also determines whether a user who is entering the system has supplied a valid operator identification card (OIDCARD) and is authorized to access the terminal. During IMS and CICS signon processing, the VERIFY request determines whether a user who is entering the system is authorized to use IMS or CICS and to access the terminal.

If the user ID, password, operator identification card, group name, terminal, and application are accepted, RACF builds an accessor environment element (ACEE) for the user.

Note: When no user ID, group, and password are passed to RACROUTE REQUEST=VERIFY, RACROUTE builds a default ACEE containing an asterisk (*) (X'5C') for the user ID and group name and returns to the issuer of the VERIFY request with a return code of 0, indicating a successful completion.

The ACEE identifies the scope of the user's authorization that will be used during the current terminal session or batch job. You can use the RACROUTE REQUEST=VERIFY(X) exit routine to supply a user ID for undefined users or to perform additional authorization checks for users. Many of the values passed to the RACROUTE REQUEST=VERIFY(X) preprocessing and postprocessing exits are

derived from the parameters specified on the RACROUTE macro. For more details, see *z/OS Security Server RACROUTE Macro Reference*.

Preprocessing exit (ICHRIX01)

The RACROUTE REQUEST=VERIFY(X) preprocessing exit routine must be named ICHRIX01. It gets control before:

- User identification
- User verification
- Terminal authorization checking

and can get control many times during one job.

This exit must be reentrant and is invoked in supervisor state, under protection key 0, with no locks held.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

z/OS Security Server RACF Data Areas contains a mapping of the VERIFY(X) request exit parameter list, RIXP.

When a started task is being verified, if the installation has included a started-procedure name in the installation's started procedures table (ICHRIN03), or an appropriate profile in the STARTED class, the VERIFY(X) request will have already converted the started-procedure name to a user ID and, optionally, a group name before performing REQUEST=VERIFY(X) processing.

When an ACEE that has a third-party ACEE attached is deleted, the RACROUTE REQUEST=VERIFY(X) request preprocessing and postprocessing exits get control for both the third-party ACEE and the original ACEE being deleted. This allows explicit access to the installation work area's ACEEIEP field for any third-party ACEEs. RACROUTE REQUEST=VERIFY(X) should not be bypassed for these unless the exit is maintaining the ACEE; that is, the exit should not leave any ACEEs in storage. The calls to the exits are nested. For example, the preprocessing exit is called for the main ACEE. Then another RACROUTE REQUEST=VERIFY(X) calls the preprocessing exit and postprocessing exit for the third party, followed by a call to the postprocessing exit for the main ACEE.

Return codes from the RACROUTE REQUEST=VERIFY(X) preprocessing exit

When the RACROUTE REQUEST=VERIFY(X) preprocessing exit routine returns control, register 15 should contain one of the following return codes:

Code	Meaning
0	Exit-routine processing is complete; normal processing is to continue.
4	The request is not accepted and is to be failed. The postprocessing exit is still invoked.
8	The request is accepted. Processing stops, but the postprocessing exit is still invoked.

Note: If register 15 contains any other value, RACROUTE REQUEST=VERIFY(X) issues an abend code (383) that indicates a nonvalid exit return code.

Do not confuse codes from the RACROUTE REQUEST=VERIFY(X) preprocessing exit routine with the return codes from the RACROUTE REQUEST=VERIFY(X) macro, which are documented in *z/OS Security Server RACROUTE Macro Reference*.

When the VERIFY(X) request preprocessing exit routine sets a return code of 8 and the caller specified ENVIR=CREATE, the exit is responsible for building an ACEE. In this case:

- Your exit routine can use the ACEE passed as input, or it can obtain its own storage for the ACEE. If the exit routine obtains its own storage, the exit must free the passed ACEE and the tables chained to it.
- It is not feasible for the exit routine to build its own ACEE directly, because the exit cannot determine the correct values for some fields in the ACEE. If you cannot use the ACEE RACF has provided, you should consider issuing RACROUTE REQUEST=VERIFY,ENVIR=CREATE and omitting the user ID, group name, and password so that RACF creates an ACEE for an unidentified user. Then you can modify that ACEE to have the values you want. You should leave ACEECGRP set to 0. Be careful that your exit recognizes your call and doesn't loop. One way to do this is to use the INSTLN keyword on the RACROUTE request.

Postprocessing exit (ICHRIX02)

The RACROUTE REQUEST=VERIFY(X) postprocessing exit routine must be named ICHRIX02. It gets control after:

- User identification
- User verification
- Terminal authorization checking

and can get control many times during one job.

This exit must be reentrant and is invoked in supervisor state, with protection key 0, with no locks held.

The exit can have any RMODE, but AMODE should be AMODE(31) or AMODE(ANY) for the best use of virtual storage.

When the RACROUTE REQUEST=VERIFY(X) postprocessing exit routine receives control, RACF has already performed the main function (for example, ACEE creation and statistics recording), but has not written any SMF records or issued any ICH408I messages.

Changes you make to the database in the postprocessing exit are not reflected in the ACEE until the next RACROUTE REQUEST=VERIFY. We recommend that you make database updates in the preprocessing exit. If you must update the RACF database in the postprocessing exit, consider using one of the following approaches to ensure that the ACEE is correct:

- After the exit updates the database, return to the RACROUTE REQUEST=VERIFY with a return code of 4, indicating a retry. This ensures that the ACEE is rebuilt with the updated information.
- Update the ACEE directly with the same update made to the database. For example, if the exit updates INSTDATA in the database, it should also update ACEEINST in the ACEE. This ensures that the current ACEE matches the database, and that a refreshed copy of the ACEE is placed in VLF if the IRRACEE VLF class is active.

z/OS Security Server RACF Data Areas contains a mapping of the RACROUTE REQUEST=VERIFY(X) exit parameter list, RIXP.

Return codes from the RACROUTE REQUEST=VERIFY(X) postprocessing exit

When the RACROUTE REQUEST=VERIFY(X) postprocessing exit routine returns control, register 15 should contain one of the following return codes:

Code	Meaning
0	Continue with RACROUTE REQUEST=VERIFY(X) processing. If the exit routine changes the values of the return code or the abend code (from zero to a nonzero value), REQUEST=VERIFY(X) uses the changed values.
4	Try the RACROUTE request again; invoke the REQUEST=VERIFY(X) preprocessing exit routine. Any values in the return- or abend-code fields are ignored, and the fields are reset to zero. Other fields are not affected. In particular, the INSTLN value is not reinitialized; this preserves any information placed in it by the preprocessing or postprocessing exit routine.

Note: If register 15 contains any other value, RACROUTE REQUEST=VERIFY(X) issues an abend code (383) that indicates a non-valid exit return code.

The REQUEST=VERIFY(X) macro may have updated the user's entry with new password information. In this case, attempts to retry the RACROUTE request without adjusting the input parameters accordingly may cause REQUEST=VERIFY(X) failure.

Do not confuse return codes from the RACROUTE REQUEST=VERIFY(X) postprocessing exit routine with the return codes from the RACROUTE REQUEST=VERIFY(X) macro, which are documented in *z/OS Security Server RACROUTE Macro Reference*.

RACF report-writer exit

ICHRSMFE is an optional, installation-written exit routine that you can use to:

- Create additional selection and rejection criteria for records that the RACF report writer processes
- Modify data set naming conventions in records that the RACF report writer processes
- Create additional output reports, in addition to the reports that the RACF report writer provides

To avoid an unresolved external reference from the link editor, ICHRSMFE is shipped as a dummy module (BR 14) that is link-edited into the RACF report-writer load module, RACFRW. To replace this dummy exit with your own, use an SMP/E user modification.

Each time the RACF report writer reads an SMF record, it calls ICHRSMFE. If the record is not a RACF SMF record, the RACF report writer calls ICHRSMFE *before* it applies record-selection criteria (from the SELECT and EVENT subcommands) to the record. If the record is a RACF SMF record, the RACF report writer calls ICHRSMFE both *before* and *after* it applies the record-selection criteria. In addition, the RACF report writer calls ICHRSMFE when it encounters end-of-file on the SMF data set.

For more information on the RACF report writer, see *z/OS Security Server RACF Auditor's Guide*.

ICHRSMFE processing

The exit need not be reentrant, but must written as though the module were serially reusable, because it may be called multiple times. The exit is invoked in problem state key 8.

z/OS Security Server RACF Data Areas contains a mapping of the RACF report-writer exit parameter list, RSMXP.

Return codes from the RACF report-writer exit (ICHRSMFE)

When the ICHRSMFE exit routine returns control to the RACF report writer, register 15 should contain one of the following return codes:

Code	Meaning
0	Exit-routine processing is complete. Normal processing is to continue.
4	Override the selection criteria and select this record.
8	Override the selection criteria and reject this record.

Note: If register 15 contains any other value, processing proceeds as if the return code were 0.

SAF router exits

The system authorization facility (SAF) and the SAF router are present on all MVS systems, even if RACF is not installed. Although the SAF router is not part of RACF, many system components and programs invoke RACF through the RACROUTE macro and SAF. Therefore, installations can modify RACF parameter lists and do customized security processing within the SAF router.

Information on the SAF router exits (ICHRTX00 and ICHRTX01) can be found in *z/OS Security Server RACROUTE Macro Reference*. Information on the SAF callable services router exit (IRRSXT00) can be found in *z/OS Security Server RACF Callable Services*.

Chapter 9. Recovery Procedures

Overview . . . 304
Exit Routine Considerations . . . 304
TSO Considerations . . . 305
The RVARY Command . . . 305
Shared Database Considerations . . . 305
RVARY Password Considerations . . . 305
Quiescing Database I/O Activity . . . 306
RVARY SWITCH . . . 307
RVARY ACTIVE or INACTIVE . . . 307
Synchronization Considerations . . . 307
Restoration of the RACF Database . . . 307
Restoration of a Single Data Set in the Database . . . 308
Other Recovery Considerations . . . 308
Considerations for Issuing RVARY from the RACFRCVY Started Procedure 308
Failures on the RACF Database . . . 309
Sample Recovery Procedures . . . 309
The Primary Database Is in Error, the Backup Database Is Unaffected 310
The Backup Database Is in Error, the Primary Database Is Unaffected 310
The Primary Database Is in Error, There Is No Backup Database . . . 310
Both the Primary and the Backup Databases Are in Error . . . 311
Failures Using Sysplex Data Sharing . . . 311
Read-Only Mode . . . 311
Non–Data Sharing Mode . . . 312
Recovery Scenarios . . . 312
Coupling Facility Not Available . . . 312
Structure Not Defined in Policy . . . 312
Structure Too Small . . . 313
Link Failure . . . 313
RACF Structure Failure . . . 314
RACF Support of the Rebuild Interface . . . 314
Sysplex recovery scenarios that require XCF-local mode . . . 314
Sysplex recovery scenarios requiring a member to be brought up with sysplex communication mode and data sharing mode inactive . . . 315
Failures during RACF Command Processing . . . 315
Commands That Do Not Modify User-Created RACF Profiles . . . 316
Commands That Have Recovery Routines . . . 316
Commands That Perform Single Operations . . . 317
Commands That Perform Multiple Operations . . . 318
Recovering from Errors in Identity Mapping Profiles . . . 319
Missing Identity Mapping Profile . . . 320
User ID Associated with an Identity Mapping Profile Does Not Exist 320
Profile Mismatch . . . 320
Recovering from Errors with Application Identity Mapping . . . 321
Mapping Profile Exists . . . 321
Missing Alias Index Entry . . . 322
User or Group Associated with an Alias Index Entry does not Exist . . . 322
Profile and Alias Index Mismatch . . . 322
Commands That Are Propagated for RACF Sysplex Communication . . . 322
Failures When Propagating RVARY Commands . . . 322
Failures When Propagating SETROPTS Commands . . . 323
Failures during RACF Manager Processing . . . 324
Failures during System Operations on RACF-Protected Data Sets . . . 325
Failures during SCRATCH or DELETE . . . 326

© Copyright IBM Corp. 1994, 2004

Failures during ALLOCATE or DEFINE 326
Failures during RENAME or ALTER 326
Failures during EOV (Non-VSAM) 327
Failures in the RACF Subsystem Address Space 327
Recovering from RACF Parameter Library Problems 327
Recovering When a Task Stops 328
Recycling an RRSF Connection 329
Recovering from VSAM Errors on the RRSF Workspace Data Sets 329
Viewing the Workspace Data Sets 329
Recovering When the Workspace Data Sets Fill Up 329
The Last Resort—Shutting Down the RACF Subsystem Address Space 331

Overview

For most activity against the RACF database, RACF uses the primary database exclusively, because most activity requires only that information be read from the database. When profile changes are made (like those resulting from RACF commands), RACF updates both the primary database and, if it is active, the backup database. Statistics can be recorded on both databases. Authorization checking, user verification (except for recording such things as INITSTATS and password changes), and the listing options use only the primary database.

Problems with the RACF database are unlikely. Nevertheless, it is helpful to have a plan of action thought out beforehand. If you believe that your RACF database contains errors, there are several things to consider doing, depending on the severity of the errors.

For minor error conditions (errors not severely affecting your system), consider running the RACF database verification utility program, IRRUT200. This utility can be used to identify inconsistencies in the internal organization of the database. For more information, see “RACF database verification utility program (IRRUT200)” on page 218.

If running IRRUT200 does not identify the problem, you might want to run the RACF database unload utility program, IRRDBU00. This utility can be used to identify the profile in error. For information on IRRDBU00, see *z/OS Security Server RACF Security Administrator's Guide*.

After running either IRRUT200 or IRRDBU00, first try to use RACF commands to fix the error. If that fails, you might need to use BLKUPD to modify your RACF database. For more information on BLKUPD, see *z/OS Security Server RACF Diagnosis Guide*.

For more severe errors and depending on your system configuration, use the RVARY command. It can be used to switch, activate, or deactivate the RACF database. For several sample recovery procedures, see “Failures on the RACF Database” on page 309.

Exit Routine Considerations

Before attempting recovery from RACF failures, an installation should review the processing performed by any active RACF exit routines to determine whether the exits are obscuring the failures. For example, when command exits are being used to modify or eliminate the standard RACF naming convention, RACF error messages might specify qualifiers supplied by the exits rather than the high-level qualifiers of the data set names.

TSO Considerations

Your installation can place all TSO logon information in the RACF database, thus eliminating the SYS1.UADS data set. However, if your installation deletes SYS1.UADS entirely, and subsequently deactivates RACF with RVARY or at IPL time, no user can log on to the system. To avoid this possibility, your installation should keep at least one user ID (with known password) in the SYS1.UADS data set. This is further discussed in “Sysplex recovery scenarios that require XCF-local mode” on page 314.

The RVARY Command

With the RVARY command you can switch, activate, or deactivate RACF databases without an IPL. You can also list the current configuration of RACF databases, or, if RACF is enabled for sysplex communication, change between data sharing and non–data sharing modes.

During recovery, you want to keep the primary database active and not go into failsoft processing. For that reason, we recommend that you use RVARY SWITCH rather than RVARY INACTIVE.

You can enter this command from an active TSO session, from a batch job, from a started task that executes the TSO terminal monitor program (TMP), or as an MVS operator command if the RACF subsystem is active.

If your RACF database has multiple data sets, you can use the DATASET operand on the RVARY command to specify which one you want switched, deactivated, or reactivated.

Shared Database Considerations

The use of the RVARY command becomes more complex when the RACF database is shared with other systems. As a general rule, all systems must be synchronized with respect to the RACF database configuration.

If your database is being shared by several systems and one of the systems stops using the primary database by issuing RVARY SWITCH or RVARY INACTIVE, *all* of the systems sharing the database must do the same thing, or the results will be unpredictable. Therefore, if you issue the RVARY SWITCH or RVARY INACTIVE command on one system, you must issue it on every other system sharing the database.

Note: If RACF is enabled for sysplex communication, it propagates the RVARY SWITCH and RVARY INACTIVE commands for you from the system on which they are entered to the other members in the data sharing group. Therefore, you only need to issue the command once.

RACF also propagates the RVARY DATASHARE and RVARY NODATASHARE commands when enabled for sysplex communication.

See *z/OS Security Server RACF Command Language Reference* for the complete syntax of RVARY.

RVARY Password Considerations

The RVARYPW operand on the SETROPTS command has two suboperands that enable a user with the SPECIAL attribute to define the passwords: SWITCH(*switch-pw*) and STATUS(*status-pw*). SWITCH(*switch-pw*) defines a password that can authorize switching the RACF database or, if RACF is enabled

for sysplex communication, changing the RACF operating mode.
STATUS(*status-pw*) defines a password to activate or deactivate RACF.

When the console operator receives the RVARY command message (ICH702A or ICH703A) requesting that the password be entered, the operator first examines the user ID to ensure that the issuer has the proper authority to enter the command. If so, the operator then enters the installation-defined password to allow the request to complete—switch, activate, or deactivate the RACF database, or change the RACF operating mode.

If your installation chooses not to provide password protection for RVARY, the operator must enter YES to allow RVARY to complete.

An installation can choose not to give the operator the passwords, but rather to keep the passwords under the control of the security administrator. The security administrator can then give the operator the passwords when necessary. After the operator receives a password, the security administrator should then change the password for security purposes.

For recovery actions to take place when the installation-defined password is not available or has been lost or destroyed, RVARY allows you to use the default password YES in some cases. RACF accepts both the default password and the installation-defined password if the RVARY was issued as an operator command from a console with master authority and the ACTIVE, NODATASHARE, or SWITCH function was requested.

Quiescing Database I/O Activity

RVARY INACTIVE DATASET, SWITCH, DATASHARE, and NODATASHARE require that RVARY quiesce RACF database I/O activity before proceeding. If any database I/O activity is in progress while the status of the database is changed, the database could get corrupted. To quiesce database activity, RVARY obtains an exclusive system-wide ENQueue, specifying *qname* as SYSZRAC2 and *rname* as each data set name affected by the RVARY command.

RVARY might also obtain:

- A SYSTEMS wide exclusive RESERVE (when RACF is not enabled for sysplex communication)
- A SYSTEMS wide shared ENQueue (when RACF is enabled for sysplex communication)

These specify *qname*=SYSZRACF and *rname*=*dataset-name* for each data set affected by the RVARY command.

All database I/O activity is done through the RACF manager, which uses the same RESERVE or ENQueue serialization techniques (exclusive for write or shared for read). Therefore, when RVARY obtains its ENQueues or RESERVE, it has established that all previous database I/O activity has completed and RVARY processing can proceed.

If I/O errors or other problems prevent the I/O database activity from completing, RVARY cannot proceed. You need to correct the problem, perhaps by cancelling jobs or users. If the DASD device containing the database has failed and is not responding, you might need to force the DASD device offline. If any of these steps are necessary, you should validate the integrity of the database by running the IRRUT200 and IRRDBU00 utilities.

RVARY SWITCH

If you have a backup database specified in the data set name table (ICHRDSNT), you can enter the RVARY SWITCH command to switch from the failing primary database to the backup database.

Before entering an RVARY SWITCH, you must ensure that the backup database is active. The SWITCH option of the command deactivates the current primary database and causes RACF to use the backup copy as the new primary. The current primary is also deallocated. You should repair the original primary and activate it at the earliest opportunity.

When an RVARY SWITCH command is issued, the database buffers are also switched. When RACF *is not* enabled for sysplex communication, it associates a set of buffers with the new primary database (the original backup database) and disassociates the buffers from the original primary database (the new backup database).

When RACF is enabled for sysplex communication and an RVARY SWITCH command is issued, RACF switches the buffers by associating the larger set of buffers with the new primary database (the original backup database) and the smaller set of buffers with the new backup database (the original primary database). After the RVARY SWITCH, the coupling facility structures associated with the original primary are associated with the new primary, and the ones associated with the original backup are associated with the new backup.

Attention

When you enter RVARY SWITCH so you can use your current backup as your new primary database, your new backup database is automatically deactivated and deallocated; therefore, you must enter the RVARY ACTIVE command to reallocate and reactivate the new backup database.

RVARY ACTIVE or INACTIVE

Without a Backup Database: If your installation does not have a backup database specified in the data set name table (ICHRDSNT), and you need to deactivate the primary database, you must use the RVARY INACTIVE command. If your database has a single data set, this puts RACF into failsoft processing. If you have multiple data sets and only some are active, you are likely to experience abends.

With a Backup Database: When you deactivate a current primary RACF database, RACF does not use the backup database, even if the backup is active. For this reason, RVARY SWITCH is recommended. You can deactivate the backup database and still keep the corresponding primary active.

Synchronization Considerations

The occurrence of DASD errors might cause synchronization problems between the RACF database and the contents of VTOCs and catalogs. You can minimize these problems by using generic data set profiles or by using discrete data set profiles in combination with an active backup RACF database.

Restoration of the RACF Database

If it becomes necessary to restore a RACF database from tape, in most cases a resynchronization is necessary before the system can be available for normal

processing again. If the changes are also recorded on SMF, the SMF data for the period between the time of the dump and the loss of the RACF data can be helpful. A program to process the RACF SMF records and create the commands necessary to update the RACF database would be useful in conjunction with manual checks.

Restoration of a Single Data Set in the Database

To reduce contention, you might have divided your database into multiple data sets. Should you have to restore one of those data sets, the synchronization problem is limited. A manual procedure might be appropriate to correct the RACF definitions for that data set. A TSO command procedure might be useful to analyze discrepancies.

Other Recovery Considerations

You can use RACF commands to accomplish all the synchronization steps (some might require the SPECIAL attribute). Zap and similar programs are unnecessary, and you should not use them. Similarly, use of the BLKUPD command should not be necessary. You should only use BLKUPD in the unlikely event that other mechanisms fail.

You should test all procedures for switching and creating or restoring copies of the RACF database before using RACF in production.

Considerations for Issuing RVARY from the RACFRCVY Started Procedure

You might find the following information helpful if you are running without the RACF subsystem. (If you have the RACF subsystem installed, you can enter the RVARY command as an MVS operator command. In that case, the started procedure described below is not needed.)

The RVARY command is normally executed from a TSO session. However, should the database become disabled, in some cases, no TSO user can logon to enter the RVARY command. To circumvent this situation, the installation can establish an alternative recovery environment by means of the RACFRCVY (RACF recovery) started procedure.

Note: To set up this started procedure, you must have TSO/E installed. The installation can implement the procedure in the following way:

Before IPL, the system programmer should do the following:

1. Ensure that the programs that constitute the RACFRCVY procedure have been compiled, assembled, and placed in the appropriate libraries. (The RACFRCVY procedure with associated programs and CLIST is shipped as members RACRVRY1, RACRVRY2, and RACRVRY3 of SYS1.SAMPLIB.) In addition, ensure that the name of the procedure has been assigned a RACF user ID. (See “Associating started procedures and jobs with user IDs” on page 94 for information.)
2. Update COMMND*xx* in SYS1.PARMLIB, so it starts at IPL time, (or, for testing purposes, have the console operator start RACFRCVY after IPL).

At IPL time, when RACFRCVY is started, it issues a WTOR, with an accompanying response number, which identifies the RACFRCVY procedure to the operator. Unless there is a problem with the RACF database, the operator would not respond to this WTOR; this leaves the procedure poised, ready if needed.

Should a problem occur with the RACF database, and the operator wants to use the RACFRCVY procedure to execute the RVARY command, the operator does the following:

1. Types in the response number that was indicated on the WTOR, followed by the RVARY command and the desired operand; for example, `R num RVARY SWITCH`.
2. Depending on the operand, RVARY, executing under the control of RACFRCVY, might prompt the operator for a password. The operator enters it, and RACF completes the command.
3. After the command completes, the RACFRCVY procedure prompts the operator to enter a C (to continue the procedure) or R (to redisplay the output from the executed command as often as required). If the operator types in C, the procedure responds by asking if the operator wants another command or if the operator wants to quit. The operator can do one of three things: type in another command, type in QUIT, which ends the procedure, or type in nothing, which results in the procedure remaining in a poised state, waiting to be summoned if another RACF database problem occurs.

Note that if the operator types in QUIT, RACFRCVY ends and must be started again in order to be used for recovery purposes. To start RACFRCVY again, the console operator types in START RACFRCVY.

If you choose not to use the RACF sample and you have multiple data sets in your RACF database, you should consider having an individual started procedure to control each data set. The user ID and group names assigned to each of these procedures should be in a data set other than the data set that the procedure controls. Each PROC should be set up to issue the appropriate RVARY command when it is started by use of the TMP (Terminal Monitor Program).

Failures on the RACF Database

In the unlikely event of I/O failures against the device upon which the RACF database resides, as described in “Quiescing Database I/O Activity” on page 306, or in the case of RACF database corruption, one of the following situations might apply:

- The primary database is in error; the backup database is unaffected.
- The backup database is in error; the primary database is unaffected.
- The primary database is in error; there is no backup database.
- Both primary and backup databases are in error.

Sample recovery procedures are provided below for each situation.

In the event of a failure due to insufficient space in the RACF database, you can use the IRRUT400 utility to copy the data set having the problem to a larger data set, or, if fragmentation alone is the problem, to another data set the same size. As the utility copies the data set, it rebuilds it and repairs any fragmentation. See “Monitoring the usable space in your RACF database” on page 13 for information on how to foresee and prevent an “insufficient space” condition.

Sample Recovery Procedures

If you have split your database and only one data set in the database is in error, only the broken data set must be recovered. When you issue the RVARY command, name the broken data set using the DATASET operand. In general, do not let the data set name default. By using the DATASET operand, you avoid accidentally processing the wrong data set.

The Primary Database Is in Error, the Backup Database Is Unaffected

In this situation, follow this procedure:

1. Ensure that the backup is active.
2. Issue RVARY SWITCH (the backup is now the *new primary*).
3. Do one of the following:
   a. Correct the problem on the original primary, using BLKUPD.
   b. If the device is accessible, copy the backup (*new primary*) onto the original primary, using IRRUT200 or IRRUT400.
   c. If the device is inaccessible, allocate, catalog and copy a replacement primary onto a different DASD device, using IRRUT200 or IRRUT400. You must catalog it on all systems sharing the database.
4. Issue RVARY ACTIVE for the original primary or replacement primary.
5. If you used step 3b with IRRUT400 with LOCKINPUT, you should now run IRRUT400 with UNLOCKINPUT to unlock your new primary (*original backup*).
6. Issue RVARY SWITCH.
7. Issue RVARY ACTIVE for the backup (*original backup*).

Note: After an RVARY SWITCH when your backup is inactive, your primary and backup databases might become out of synch. If this is a concern to you, the safest approach is to use step 3b, and use IRRUT400 with LOCKINPUT. But note that even in this scenario your databases could become out of synch between steps 6 and 7.

The Backup Database Is in Error, the Primary Database Is Unaffected

In this situation, follow this procedure:

1. Issue RVARY INACTIVE for the backup.
2. Do one of the following:
   a. Correct the problem on the backup, using BLKUPD.
   b. If the device is accessible, copy the primary onto the backup, using IRRUT200 or IRRUT400.
   c. If the device is inaccessible, allocate, catalog, and copy a replacement backup onto a different DASD device, using IRRUT200 or IRRUT400.
3. Issue RVARY ACTIVE for the backup.
4. If you used step 2b with IRRUT400 with LOCKINPUT, you should now run IRRUT400 with UNLOCKINPUT to unlock your new primary (*original backup*).

Note: To minimize the possibility of your primary and backup databases getting out of synch, the safest approach is to use step 2b with IRRUT400 and LOCKINPUT.

The Primary Database Is in Error, There Is No Backup Database

In this situation, follow this procedure:

1. Issue RVARY INACTIVE. Failsoft processing is in effect. See “Failsoft processing” on page 102.
2. Obtain the most recent dump of your RACF database.
3. Do one of the following:
   a. If the device is accessible, copy the dump to the primary database.
   b. If the device is inaccessible, allocate, catalog, and copy the database onto a different DASD device.

4. Issue RVARY ACTIVE.

   Your database is probably back-level. To bring it up to date, use a combination of the SMF records and the RACF report writer to add or delete the appropriate profiles and access authorities.

Both the Primary and the Backup Databases Are in Error

In this situation, follow the procedure for the situation in which your primary database is in error and you have no backup database.

Once you have the primary database, follow the procedure for the situation in which the backup database is in error and the primary database is unaffected.

Failures Using Sysplex Data Sharing

When a problem with the coupling facility prevents RACF from entering data sharing mode, RACF provides several alternatives for recovery. See *z/OS Security Server RACF Diagnosis Guide* for information about sysplex recovery.

Read-Only Mode

A system experiencing a problem using one or more RACF cache structures might enter read-only mode, with RACF issuing message IRRX004I. With the exception of statistics updates during logon and job initiation, and other statistics updates made with ICHEINTY ALTERI requests, the RACF manager rejects requests to update the RACF database with return code X'50'. This might cause occurrences of abends 483-50 or 485-50.

It is important to note that error messages resulting from read-only mode indicate a coupling facility problem, and not a problem with the RACF database. Along with message IRRX004I, RACF issues one or more additional diagnostic messages to assist in correcting the problems with the coupling facility.

Because serialization for read-only mode is compatible with serialization for data sharing mode, other systems in the sysplex can continue using the coupling facility while the operator corrects the problem on the read-only system.

RVARY and SETROPTS commands that are propagated to systems in read-only mode run on the read-only system; however, only SETROPTS LIST and RVARY commands can be issued on a system that is in read-only mode. If you must update the RACF database and one or more systems are in read-only mode:

1. Attempt to make the update from a system in data sharing mode.
2. If all systems are in read-only mode and you do not want to wait for the coupling facility problem to be fixed, you can issue the RVARY NODATASHARE command to switch into non–data sharing mode. This allows updates to be made to the RACF database.

While in read-only mode, RACF listens for ENF signals indicating any changes in coupling facility availability and automatically tries to connect and enter data sharing mode when it receives an ENF signal that indicates that RACF-related resources are available. This might result in repeated IXLCONN failures due to unrelated coupling facility changes. For this reason, RACF does not issue error message IRRX003A when the connect attempt is due to an ENF signal. MVS does, however, log every IXLCONN failure in SYS1.LOGREC, so that if multiple errors occur diagnostics are available.

Non–Data Sharing Mode

Systems in non–data sharing mode have full read/write capability to the RACF database, although they do not use the coupling facility. RACF uses hardware RESERVEs to serialize access to the RACF database (unless the installation has explicitly converted the RESERVES to global ENQs using global resource serialization). Either way, non–data sharing mode is not compatible with data sharing or read-only modes; if one system in the RACF data sharing group is in non–data sharing mode, they all are in non–data sharing mode. This mode is used when the installation wants sysplex communication enabled and does not want to use a coupling facility, but it can also be useful in certain coupling facility recovery operations.

While in non–data sharing mode, RACF ignores ENF signals, and does not automatically try to enter data sharing mode.

If the coupling facility is available and you want to enter data sharing mode, issue the RVARY DATASHARE command. If you do this when the coupling facility is not available, you enter read-only mode.

Recovery Scenarios

Several recovery scenarios follow for failures that can occur while using RACF sysplex data sharing.

Coupling Facility Not Available

If the coupling facility is not online or the system does not have connectivity to it, RACF issues message IRRX003A (with accompanying return and reason codes) for each structure, and the system is initialized in read-only mode. If the problem is only connectivity, it might be specific to this system (other systems might be initialized in data sharing mode).

It is possible that the installation had no plans to use the coupling facility but, in setting up ICHRDSNT for sysplex communication, accidentally turned on the data sharing mode bit.

The operator has three choices:

- If data sharing was actually intended, leave affected systems in read-only mode temporarily. Bring the coupling facility online or fix any link problems. Through ENF signaling, RACF is notified of the coupling facility's availability. Then RACF automatically reattempts connecting to structures so that it can enter data sharing mode.
- If data sharing was intended, and at least one system still has connectivity to the structure, then the structure can be rebuilt into another coupling facility to which more systems have connectivity. See "RACF Support of the Rebuild Interface" on page 314 for more information.
- Issue RVARY NODATASHARE to put the sysplex into non-data sharing mode. If the coupling facility was not wanted and the problem is simply in the ICHRDSNT bit settings, be sure to remedy this in case the system needs to be re-IPLed in the future. Otherwise, fix the coupling facility problem as previously mentioned, then issue RVARY DATASHARE; this causes RACF to reattempt connecting to structures so that the system can enter data sharing mode.

Structure Not Defined in Policy

An installation might not have all of its RACF structures defined in the active policy at the time a sysplex is IPLed. RACF issues message IRRX003A (with accompanying return and reason codes) for each missing structure, and all systems

are initialized in read-only mode. Update the policy, and RACF will automatically reattempt connecting to structures through ENF signalling.

Structure Too Small

RACF requires that there be at least as many cache structure entries as there are local buffer slots for the associated data set. If this is not true, RACF remains connected to that structure but enters read-only mode. This will probably affect every system in the RACF data sharing group, but there might be exceptions if different systems use a different number of local buffers for the same data set. Regardless, there are several possible reasons for the cache structure being too small:

- The requested size for the RACF structure in the MVS policy is too small. In this case, IRRX011A is issued.
- The policy definitions specific to the RACF structures meet the minimum requirements, but the total of all the structures' policy definitions is greater than the amount of space available in the coupling facility. If the resulting size of a RACF structure still meets the RACF minimum, only message IRRX012I is issued, and the system can still enter data sharing mode. If not, IRRX012I and IRRX013A are issued and the system enters read-only mode as in the previous case.
- The policy's STRUCTURE statement specifies the INITSIZE keyword. The STRUCTURE statement should not specify INITSIZE, because RACF does not support the ALTER function of coupling facility structures. Specifying INITSIZE causes the size of the structure to be limited to the INITSIZE value instead of the SIZE value. If the INITSIZE value is less than the SIZE value, RACF issues an informational message, IRRX012I.

  IF RACF issues only message IRRX012I, the structure size (that is, the INITSIZE value) meets the RACF minimum, and the system can still enter data sharing mode. If IRRX012I is followed by message IRRX013A, the structure size is too small and the system enters read-only mode.

If the policy needs to be changed, the operator must decide whether to change RACF's mode across the sysplex while that change is being made. The choices are:

- Leave the system in read-only mode while the policy is being corrected, then issue the MVS SETXCF START,REBUILD operator command to rebuild the structures so that they reflect the new policy. See "RACF Support of the Rebuild Interface" on page 314 for more information.
- Issue RVARY NODATASHARE to put the sysplex into non–data sharing mode. This also causes RACF to disconnect from all structures. Because they are non-persistent, the structures are taken out of the coupling facility. Next update the policy and issue the SETXCF command to start the policy. After the policy has been started with the RACF-related modifications, issue the RVARY DATASHARE command; this causes RACF to reattempt connections so that the system can enter data sharing mode.

Link Failure

This problem is the equivalent of the problem scenario "Coupling Facility Not Available" on page 312 in terms of system connectivity, but the problem occurs after RACF has already been in data sharing mode, which affects how MVS and RACF react to the problem. Typically, the link failure is detected when RACF attempts to make use of the coupling facility via the IXLCACHE macro; IRRX016I is issued. One of two recovery actions occur automatically:

1. If REBUILDPERCENT was specified in the coupling facility resource management (CFRM) policy for the RACF structure such that the percentage of

system-weight losing connectivity has exceeded the limit, MVS initiates a rebuild for that structure. See “RACF Support of the Rebuild Interface” for more information.

2. MVS decides not to initiate a rebuild. MVS makes this decision if, for example, REBUILDPERCENT was not specified in the CFRM policy for the RACF structure, a sysplex failure management policy is not active for specifying the system-weights, or the system-weights were specified but the loss-of-connectivity threshold was not reached. In all cases MVS notifies RACF that a rebuild is not being done, and RACF disconnects the system from the problem structure and issues message IRRX015I. If the system is a data sharing system, it enters read-only mode. If the structure is deallocated from the coupling facility via disconnection from all connectors, RACF attempts to connect to a structure in an alternate coupling facility, if available.

RACF Structure Failure

A structure failure can occur after RACF is already connected to structures. RACF issues message IRRX020I to indicate that a rebuild has been initiated. See “RACF Support of the Rebuild Interface” for more information.

RACF Support of the Rebuild Interface

RACF supports the rebuild interface. See *z/OS MVS Programming: Sysplex Services Guide* for information on the rebuild interface. RACF issues message IRRX020I to indicate that a rebuild has been initiated, and issues message IRRX008I upon its completion. A rebuild can be initiated due to:

- Link failure
- Structure failure
- A SETXCF operator command
- An authorized program issuing the ?IXLREBLD START macro

If problems are encountered, one or more of the following messages are issued:

IRRX001I
IRRX002I
IRRX003A
IRRX004I
IRRX010I
IRRX011A
IRRX012I
IRRX013A

For information on these messages, see *z/OS Security Server RACF Messages and Codes*. If an alternate coupling facility is not available and problems such as link failures cannot be fixed readily, RVARY NODATASHARE can be used to move all systems into non–data sharing mode while the problems are fixed.

Note: RACF’s support of the rebuild interface requires that the support be installed on each system in the sysplex. The support is provided by:

- OS/390 Release 3 Security Server (RACF) and above
- OS/390 Release 2 Security Server (RACF) with PTF UW90293
- OS/390 Release 1 Security Server (RACF) with PTF UW90293
- RACF 2.2 with PTF UW90293
- RACF 2.1 with PTM UW90292

Sysplex recovery scenarios that require XCF-local mode

There are some sysplex recovery scenarios that require a member to be brought up in XCF-local mode. RACF will not come up when the data set name table asks for at least data-communication and the system is in XCF-local mode. (This is called failsoft -- see “Failsoft processing” on page 102). RACF is designed this way

because there is a significant possibility of RACF database corruption when 1 member is up in XCF-local mode and other member(s) are up in sysplex communication or data sharing mode.

We recommend that sysplex customers use the TSO user attributes dataset (UADS) to authorize at least one ID. This gives you a way to log on to TSO. But the system will have limited functionality due to the absence of RACF.

If a user logs on to TSO and you have not defined a TSO segment for that user, TSO checks the SYS1.UADS data set for the information it needs to build a session. If TSO does not find an entry for the user in SYS1.UADS, the user is denied access to the system. You must maintain entries in SYS1.UADS for emergency use (at least IBMUSER and and one system programmer is recommended). This allows you to log on to TSO when RACF is not up (failsoft). (Failsoft is documented further in “Failsoft processing” on page 102).

It is better to define UADS emergency IDs without RACF TSO segments. If you have a TSO segment, information from there (such as password) is used to log on when RACF is active. However during an emergency when RACF is not up, UADS information is used (which might not be the same).

TSO provides the UADS mechanism to allow users to be defined that can log on in this situation. If one or more IDs are not set up this way, in advance, there is no way to log on to any TSO user ID should you need to bring a member up in XCF-local mode.

For further information on UADS, refer to:

- *z/OS TSO/E Administration*
- *z/OS TSO/E Customization*
- *z/OS TSO/E System Programming Command Reference*

Sysplex recovery scenarios requiring a member to be brought up with sysplex communication mode and data sharing mode inactive

There are some recovery scenarios that require that one member be brought up with sysplex communication mode and data sharing mode inactive in order to accomplish recovery actions. This can be done utilizing an ″emergency ICHRDSNT″ with both the sysplex communication and data sharing bits off. “Emergency data set name tables” on page 40 describes how to set up such an emergency ICHRDSNT.

This will allow a single member to be brought up with approximately full functionality, for the sake of accomplishing recovery actions.

Failures during RACF Command Processing

System or RACF failures that occur during the processing of RACF commands can cause discrepancies between the various profiles on the RACF database. (For example, a failure during ADDUSER command processing can result in the user profile being created but the default group profile not being updated with the new user ID.)

In this section, the RACF commands are grouped in categories based on the operations the commands perform on the RACF database.

Notes:

1. If RACF is running in read-only mode, you might see error messages that appear to indicate DASD problems, but are actually caused by problems with the coupling facility. See “Failures Using Sysplex Data Sharing” on page 311.
2. The operator must have a specific authority to enter some command operands. See *z/OS Security Server RACF Command Language Reference* for more information about these commands and their operands.

Commands That Do Not Modify User-Created RACF Profiles

The commands that do not modify user-created RACF profiles are:

- DISPLAY
- LISTDSD
- LISTGRP
- LISTUSER
- RESTART
- RLIST
- RVARY
- SEARCH
- SET (except SET INCLUDE)
- SETROPTS
- SIGNOFF
- STOP
- TARGET

Note: If the RACGLIST class is active, and a RACGLIST *classname* profile exists on the database, SETROPTS RACLIST(*classname*) and SETROPTS RACLIST(*classname*) REFRESH will create or modify RACGLIST profiles.

Failures that occur during the processing of these commands do not cause problems with the profiles on the RACF database because these commands do not modify profiles. However, the SETROPTS command does rewrite the inventory control block (ICB) in the primary RACF database.

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. Reenter the command.
3. If the failure occurs again, contact the IBM support center.

Commands That Have Recovery Routines

Failures that occur during the processing of the following commands might or might not cause a problem with the profiles on the RACF database. These commands have recovery (backout) routines that enable the command processor to recover from some of the failures.

The commands are:

- ADDGROUP
- ADDUSER
- ALTGROUP
- CONNECT

If the command error messages indicate that recovery (backout) was successful, perform the following steps:

1. Examine the error messages to identify the failure.
2. Reenter the command.

3. If the failure occurs again, contact your programming support representative.

If the command error messages indicate that recovery (backout) was not successful, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the contents of the affected user and group profiles to determine the status of the contents.
3. If no profiles were modified, reenter the command.
4. If the user or group profiles have discrepancies, enter the appropriate commands to correct the data in the profiles.

   Example: A failure occurs during the processing of the ADDUSER command and the user profile is created correctly but the group profile is not updated with the new user's user ID. In this case, enter the CONNECT command with the default group name as the desired group in order to update the group profile.
5. If the command was adding or changing a UID or GID of an OVM segment, and the user or group profile is correct, examine the appropriate VMPOSIX mapping profile to see if it matches the change made to the user or group profile. If it does not match, change the VMPOSIX profile appropriately.

   Example: You entered:

   ```
   ADDUSER CAMERON OVM(UID(7))
   ```

   The CAMERON user profile is correct but the U7 profile does not exist in the VMPOSIX class. Add it as follows:

   ```
   RDEFINE VMPOSIX U7 UACC(NONE)
   PERMIT U7 CLASS(VMPOSIX) ID(CAMERON) ACCESS(NONE)
   PERMIT U7 CLASS(VMPOSIX) ID(your-id) DELETE
   ```

   If the NOADDCREATOR option is in effect, the PERMIT command to delete authorization for your user ID is not necessary.

   For information on VMPOSIX mapping profiles, see *RACF Security Administrator's Guide* for RACF 1.10 for VM. For information on the NOADDCREATOR option, see *z/OS Security Server RACF Security Administrator's Guide*. For information on the ADDCREATOR and NOADDCREATOR keywords on the SETROPTS command, see *z/OS Security Server RACF Command Language Reference*.
6. If there are no discrepancies and the user and group profiles and the VMPOSIX mapping profiles (if relevant) are correct, the command completed successfully.
7. If the failure occurs again, contact your programming support representative.

Commands That Perform Single Operations

The following commands modify only one profile at a time on the RACF database. Therefore, failures that occur during the processing of these commands affect only one profile.

The commands are:

- ALTDSD (without the ADDVOL, ALTVOL, or DELVOL operand)
- PASSWORD
- PERMIT
- RALTER
- RDEFINE
- RDELETE

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the contents of the affected user or resource profile to determine the status of the contents.
3. If the requested update was not made to the user or resource profile, reenter the command.
4. If the requested update was made, the operation completed successfully before the error occurred.
5. If the failure occurs again, contact your programming support representative.

Commands That Perform Multiple Operations

The following commands perform more than one operation on the RACF database. Therefore, failures that occur during the processing of these commands can cause discrepancies between the profiles on the RACF database, or discrepancies between data set profiles and the RACF-protected indication for the data set.

The commands are:

- ADDGROUP
- ADDSD
- ADDUSER
- ALTDSD (with the ADDVOL, ALTVOL, or DELVOL operand)
- ALTGROUP
- ALTUSER
- DELDSD
- DELGROUP
- DELUSER
- RACDCERT
- RACLINK
- REMOVE

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the contents of the affected user, group, and data set profiles, and any relevant mapping profiles (NOTELINK, NDSLINK, DCEUUIDS, UNIXMAP, and VMPOSIX) to determine the status of the contents. Do not expect to find any UNIXMAP, NOTELINK, or NDSLINK mapping profiles if your system is running with application identity mapping stage 3. Instead, run IRRUT200 to verify the alias index entries.
3. If all information is correct, the command completed successfully before the error occurred.
4. If the profiles contain incorrect information, enter the appropriate commands to correct the profiles.

   Example 1: During REMOVE command processing, a failure occurs that causes the connect entry for the user to be deleted but does not delete the user's user ID from the group profile. In this case, reenter the REMOVE command.

   Example 2: During DELUSER processing, a failure occurs that causes the user's profile to be removed, but the user ID remains in the default group. In this case, enter the CONNECT command with the REVOKE operand to remove the user ID from the default group.

   Example 3: During ADDSD command processing, a failure occurs that causes the RACF-protected indicator in the DSCB (or catalog) to be set but prevents

the creation of the data set profile. In this case, enter the ADDSD command with the NOSET operand to create the data set profile.

Example 4: During DELDSD command processing, a failure occurs that causes the RACF-protected indicator in the DSCB (or catalog) to be set off but does not delete the data set profile from the RACF data set. In this case, enter the DELDSD command with the NOSET operand.

Example 5: During ADDUSER command processing for the command:

```
ADDUSER SIVLE OVM(UID(10))
```

a failure occurs that causes the user's profile to be created without creating the corresponding U10 mapping profile in the VMPOSIX class. In this case, enter:

```
RDEFINE VMPOSIX U10 UACC(NONE)
PERMIT U10 CLASS(VMPOSIX) ID(SIVLE) ACCESS(NONE)
PERMIT U10 CLASS(VMPOSIX) ID(your-id) DELETE
```

If the NOADDCREATOR option is in effect, the PERMIT command to delete authorization for your user ID is not necessary. If another user already has a UID of 10, the VMPOSIX profile probably exists, and the RDEFINE command is not necessary. For more information on VMPOSIX mapping profiles, see *RACF Security Administrator's Guide* for RACF 1.10 for VM. For more information on the NOADDCREATOR option, see *z/OS Security Server RACF Security Administrator's Guide*. For information on the ADDCREATOR and NOADDCREATOR keywords on the SETROPTS command, see *z/OS Security Server RACF Command Language Reference*.

Example 6: During ADDUSER command processing for the command:

```
ADDUSER DCEUSR DCE(UUID(004386ea-ebb6-1ec3-bcae-10005ac90feb))
```

a failure occurs that causes the user's profile to be created without creating the corresponding 004386ea-ebb6-1ec3-bcae-10005ac90feb mapping profile in the DCEUUIDS class. In this case, enter:

```
RDEFINE DCEUUIDS 004386ea-ebb6-1ec3-bcae-10005ac90feb UACC(NONE)
 APPLDATA('DCEUSR')
```

Example 7: During ADDUSER command processing for the command:

```
ADDUSER USER0131 OMVS(UID(0))
```

a failure occurs and messages ICH51011I, ICH01010I, and IRR419I are issued, indicating that an alias index entry has reached its maximum size and no additional users can be associated with the UID. Although the user profile is created with the UID field complete, processsing failed before the mapping profile, alias index, or connect link to the default group was defined. The simplest solution is to delete the user:

```
DELUSER USER0131
```

Expect message ICH04002I even though the profile is successfully deleted. The message results from RACF's detection of the missing connect link. You can now add the user again, specifying a different UID.

5. If the failure occurs again, contact your programming support representative.

Recovering from Errors in Identity Mapping Profiles

An identity mapping profile maps an application user name to a RACF user ID if you are using generic ID mapping. If your RACF database is at application identity mapping stage 1 or higher, see "Recovering from Errors with Application Identity

Mapping" on page 321. Applications such as Lotus Notes for z/OS and Novell Directory Services for OS/390 that support RACF application identity mapping can determine the RACF user ID for a user who has been authenticated with an application user name or a digital certificate, and use the RACF user ID for authorization checking when accessing MVS resources. Identity mapping profiles for Lotus Notes for z/OS are in the NOTELINK class, and profiles for Novell Directory Services for OS/390 are in the NDSLINK class. RACF maintains these profiles during ADDUSER, ALTUSER, and DELUSER command processing. For each identity mapping profile, RACF maintains a corresponding identity segment in the USER profile: an NDS segment for Novell Directory Services for OS/390 and an LNOTES segment for Lotus Notes for z/OS. However, it is possible that an application identity mapping profile might be inadvertently deleted, or modified so that it does not match the corresponding USER profile. To correct these problems, you must administer the mapping profiles directly using RACF commands, as described in the following sections.

Note: Application user names can contain blanks, but RACF profile names cannot. When RACF creates an identity mapping profile, it replaces blank characters in the name with "¢" characters (X'4A').

For more information on identity mapping profiles, see *z/OS Security Server RACF Security Administrator's Guide*.

Missing Identity Mapping Profile

If an identity mapping profile defined in a USER profile LNOTES or NDS segment does not exist, RACF authorization checking is not able to retrieve the RACF user ID for an application user identity, and ALTUSER and DELUSER command processing cannot locate the target mapping profile. In these situations RACF issues message IRR52151I. To correct the problem:

1. Delete the identity segment (NDS or LNOTES) of the USER profile corresponding to the missing mapping profile. Do this by issuing the ALTUSER command with the NOLNOTES or NONDS operand.
2. Recreate the identity segment using the ALTUSER command with the LNOTES or NDS operand, specifying an application user name. RACF automatically creates a corresponding identity mapping profile in the NOTELINK or NDSLINK class.

User ID Associated with an Identity Mapping Profile Does Not Exist

If the user ID associated with an identity mapping profile does not exist, delete the identity mapping profile.

- If the application user name contains only uppercase characters, use the RDELETE command to delete the profile.
- If the application user name contains lowercase characters, you cannot use the RDELETE command to delete the profile. One way to delete the profile is to create a dummy user ID with an LNOTES or NDS segment specifying the application user name for the identity mapping profile you want to delete. Then delete the dummy user ID. Another way to delete the profile is to modify the RACF database using the BLKUPD command. For information on BLKUPD, see *z/OS Security Server RACF Diagnosis Guide*.

Profile Mismatch

A mismatch can occur between an identity mapping profile and the corresponding USER profile if you specify an application user name for a user, and that name has already been specified for another user. The USER profile for the second user is updated, but the identity mapping profile is not. This results in two USER profiles

pointing to the same identity mapping profile, but the identity mapping profile refers only to the first user for whom the application user name was specified. If ADDUSER or ALTUSER command processing detects a profile mismatch, it issues message IRR52154I, identifying the mapping profile and USER profile that conflict. To correct the situation:

1. Determine the first user ID that was assigned the application user name.
   - If the application user name contains lowercase letters, use the `RLIST NOTELINK *` command or the `RLIST NDSLINK *` command in the background and direct the command output to a data set. You can then use the TSO EDIT `FIND` command to locate the application user name in the data set. You can find the user ID in the application data field of the resource profile for the NOTELINK or NDSLINK class.
   - If the application user name contains only uppercase letters, issue the `RLIST NOTELINK` *`application-user-name`* or `RLIST NDSLINK` *`application-user-name`* command, using the terminal monitoring program (TMP). You can find the user ID in the application data field.
2. Issue an ALTUSER command with the NOLNOTES or NONDS operand for the first user ID to temporarily delete the user's identity mapping profile.
3. Select a new application user name for the second user ID and issue an ALTUSER command to associate the user ID with the new application user name.
4. Issue an ALTUSER command again for the first user ID and specify the user's original application user name. This command recreates the original user's identity mapping profile that was deleted in step 2.

Recovering from Errors with Application Identity Mapping

With application identity mapping enabled at stage 3, RACF uses an alias index rather than mapping profiles to associate users and groups with z/OS UNIX, Lotus Notes, and Novell Directory Service identities. It is possible that an unexpected error could cause an association mismatch that you can identify by comparing IRRUT200 alias index output with profile information returned from LISTUSER, LISTGRP, or DBUNLOAD. This section suggests methods to correct such inconsistencies.

At stages below application identity mapping stage 3, RACF maintains mapping profiles and functionality to ensure mapping compatibility with systems running RACF release 10 or below that share a database with higher-level systems. This means that the RACF database is susceptible to errors described in "Recovering from Errors in Identity Mapping Profiles" on page 319 and the recovery instructions there are equally useful. You should use program control to be sure that USER and GROUP commands can only be issued from systems running RACF release 10 or higher. After all systems sharing the database are migrated to release 10 or higher, run IRRIRA00 to advance the database to stage 3, thereby reducing the likelihood of mapping errors.

Mapping Profile Exists

If your database is at application identity mapping stage 3, no generic profiles in class UNIXMAP, NOTELINK, or NDSLINK should exist. If you find one, you can ignore it just as RACF does, or you can delete it using RDELETE. For example:

```
RDELETE UNIXMAP U1
```

If the mapping profile contains lowercase letters, you cannot specify them on the RDELETE command. You must use BLKUPD or RACROUTE to delete the profile.

If your database is at stage 0, 1, or 2, and you believe the profile to be incorrect, refer to “Recovering from Errors in Identity Mapping Profiles” on page 319 for instruction.

Missing Alias Index Entry

If your database is at stage 0, you should not expect to see any alias index entries. If your database is at a higher stage and you do not find an alias index entry corresponding to a specific UID, GID, SNAME, or UNAME, you can regenerate the entry by altering the user or group profile with the desired entry. For example:

```
ALTUSER YOURID OMVS(UID(1))
```

User or Group Associated with an Alias Index Entry does not Exist

If the profile associated with and alias index entry does not exist, you can remove the entry by temporarily adding the referenced profile with the indicated alias, then deleting the profile. For example:

```
ADDUSER YOURID OMVS (UID(1))
DELUSER YOURID
```

Profile and Alias Index Mismatch

If an alias index entry references the incorrect user or group, you can correct the index by altering the incorrect profile that references the given alias entry, altering it again to reference another alias entry, and finally altering the desired profile to reference the given alias entry. For example, if the alias index entry for UID 1 references MYID rather than the desired YOURID:

```
ALTUSER MYID OMVS(UID(1))
ALTUSER MYID OMVS(UID(2))
ALTUSER YOURID OMVS(UID(1))
```

Commands That Are Propagated for RACF Sysplex Communication

When RACF is enabled for sysplex communication, it propagates RVARY and SETROPTS commands (except RVARY LIST and SETROPTS LIST) to the other members of the RACF data sharing group.

Most SETROPTS options are propagated by updating the ICB, which is shared by all members of the RACF data sharing group. The following options are exceptions, and are propagated via XCF messaging services to the other members of the sysplex:

- RACLIST
- RACLIST REFRESH
- NORACLIST
- GENERIC REFRESH
- GLOBAL
- GLOBAL REFRESH
- WHEN (PROGRAM)
- WHEN (PROGRAM) REFRESH

The RACF data sharing group member on which a propagated command is issued is referred to as the *coordinator*. This member coordinates the propagation of the command to the other members.

Failures When Propagating RVARY Commands

Before propagating an RVARY command, RACF performs much the same initial validation that it does if it is not propagating the command. With the exception of RVARY ACTIVE and RVARY INACTIVE, the RVARY command must be valid on the RACF data sharing group member on which it is issued for the command to be

propagated. For example, if the RVARY command is issued on a member where RACF is permanently inactive, RACF issues message ICH15001I and the command does not run on the inactive member and is not propagated to any other member. Some other examples where the command fails and is not propagated are:

- The operator does not supply the correct RVARY password.
- RVARY SWITCH is specified and a required backup database is inactive.

However, if RVARY ACTIVE is issued and the target data sets are already active, or RVARY INACTIVE is requested and the target data sets are already inactive, message ICH15002I is issued, but the command is still propagated because it might be applicable on another member.

If the initial validation is successful, RACF attempts to propagate the RVARY command. However, a failure can still occur that causes RACF to not process the command. For example, when RACF attempts to propagate a command, each of the members of the data sharing group does a secondary validation of the command. If the secondary validation fails on any of the peer members, RACF issues message ICH15025I, and no member processes the command. For example, if RVARY SWITCH is issued and is valid on the coordinator, but one member has an affected inactive backup data set, RACF issues ICH15025I and the request is not processed by any member. It might be necessary to direct an RVARY LIST command to the failing member to determine the inactive backup data set. XCF communication failures can also cause a command to not be processed by any of the peer members. Check your system log for more information, such as message IRRX006I, which is issued to identify any group members which detected a RACF validation failure.

If the initial and secondary validations are successful, the coordinator requests that all peer members of the RACF data sharing group process the command. If one or more members experience a processing failure, message ICH15022I is issued. Note that in this case, unless every member experiences the same failure, the command is processed by some members. RACF issues message IRRX006I to identify which member experienced a processing failure. Check the failing member's system log for additional RACF messages that further identify the problem. Examples include message ICH556I for RACF manager invocation failures and ICH557I for a failure establishing recovery for processing the propagated RVARY command.

If the coordinator detects a severe error, it issues message ICH15026I, leaves the group, and puts itself into permanent failsoft. You must re-IPL to return this member to an active state. An example of this type of error is an abend while an RVARY SWITCH affecting multiple data sets is in progress. The member cannot continue to participate in the group with its RACF configuration in an inconsistent state, so it leaves. Note that such an error could also happen to a peer group member while processing a propagated RVARY command. In this case, messages ICH15022I and IRRX006I are issued by the coordinator and the failing member leaves the group and puts itself into permanent failsoft.

Failures When Propagating SETROPTS Commands

Before propagating a SETROPTS command, RACF performs much the same validation that it has does if it is not propagating the command. The SETROPTS command must be valid on the RACF data sharing group member where it is issued for the command to be propagated. SETROPTS commands other than SETROPTS LIST fail if the coordinator is in read-only mode.

If the validation is successful, RACF attempts to process the command on the coordinating system. If the command fails on the coordinator, RACF does not attempt to propagate the command to the peer systems.

If the coordinator can process the command successfully, RACF requests that all peer members of the RACF data sharing group process the command. A propagated SETROPTS command can run on a peer system even if the peer system is in read-only mode. If a peer system encounters an error while running the command and it is running in non–data sharing mode, the command continues to run on all systems on which it can. However, if a peer system encounters an error while running the command and it is in data sharing mode, RACF terminates the command on all systems. RACF issues message IRRX006I on the console to identify the member of the group for which the command failed. Check the failing member's system log for additional RACF messages that further identify the problem.

If XCF fails trying to propagate a command, the command might run on some members of the data sharing group but not others. When you have resolved the XCF problem, reissue the SETROPTS command.

During the processing of a SETROPTS command on a sysplex, RACF records diagnostic information in symptom records written to SYS1.LOGREC if either of the following occurs:

- Any member system in the sysplex sends or receives an emergency cancel during SETROPTS processing.

  Each member system in the sysplex processes a SETROPTS command in two phases. If an error occurs after the completion of phase one but before the completion of phase two on a member system, that system sends a message called an emergency cancel, which it propagates to other sysplex member systems. When a member system receives an emergency cancel it terminates SETROPTS processing in progress.
- The coordinator for a SETROPTS command leaves the sysplex before the command completes. The coordinator is the sysplex member system on which the SETROPTS command originated.

The IBM Support Center can help with interpretation of the symptom records during problem determination.

Failures during RACF Manager Processing

The RACF manager performs operations on the RACF database at the request of the RACF commands, RACF utility programs, and RACF SVC processing routines. Failures that occur during RACF manager processing can cause serious problems in the index entries and other records in the RACF database.

If RACF is enabled for sysplex communication, a system experiencing a problem with one or more RACF cache structures might enter read-only mode, with RACF issuing message IRRX004I. Except for statistics updates during logon and job initiation, and other statistics updates made with ICHEINTY ALTERI requests, the RACF manager rejects requests to update the RACF database with return code X'50'.

For messages IRR402I, IRR403I, and IRR404I, see *z/OS Security Server RACF Messages and Codes* for the error recovery procedures listed with each message under the heading "Problem Determination."

For messages other than IRR402I, IRR403I, and IRR404I that indicate a failure has occurred during RACF manager processing, the system programmer or security administrator performs the following steps:

1. Reenter the RACF command or RACF utility, or perform the system operation again.
2. If the failure occurs again, it is likely that you have a problem with an index entry or profile entry in your RACF database. Because the index structure is required to locate profile data, it is essential to have a valid index structure. Therefore, you should perform the following steps in order during problem determination to find the failing profile.
   a. Run the RACF database verification utility program (IRRUT200) with the INDEX and MAP ALL options to identify problems with the RACF database. For a description of the types of problems the utility finds, see the description of IRRUT200 in Chapter 7, “RACF database utilities,” on page 201.

      If IRRUT200 does not detect any problems in the RACF database structure (it verifies the index structure down to the profile level), try running the RACF database unload utility (IRRDBU00). The IRRDBU00 utility must read every profile in the database and thereby might (implicitly) identify profiles with errors. If IRRDBU00 encounters a profile in error, it might issue message IRR67092. This message contains an ICHEINTY return and reason code and also the entry name of the profile being processed.

      If you do not receive this message, but rather abend or terminate in another fashion, you might also be able to determine the profile in error. To do this, look in the output data set (OUTDD) and find the last profile (at the bottom) that was unloaded. It is likely that this profile is correct. However, the next profile in the database (in the same class) could possibly be in error, if indeed a bad profile is causing the utility to terminate.

      You can find the next profile in the database by examining the output of an IRRUT200 utility run (specifying INDEX FORMAT), or by using the BLKUPD command to examine an online database.
   b. Attempt to correct the problem using RACF commands. If this does not work, use BLKUPD to correct the problem in the RACF database.
   c. Rerun the IRRUT200 utility program to determine if there are any additional problems. If so, use BLKUPD to correct the additional problems.

For messages IRR402I, IRR403I, and IRR404I, the system programmer or security administrator should perform steps 2a and 2b.

Failures during System Operations on RACF-Protected Data Sets

Failures during system operations affect only data sets that are protected by the use of discrete profiles. (These system operations do not automatically create, modify, or delete generic profiles.)

System failures that occur during the processing of the following operations can cause discrepancies between the data set profiles in the RACF database and the indication that a data set is RACF-protected in the DSCB (for non-VSAM data sets) or the catalog (for VSAM data sets).

- SCRATCH (non-VSAM) and DELETE (VSAM)
- ALLOCATE (non-VSAM) and DEFINE (VSAM)
- RENAME (non-VSAM) and ALTER (VSAM)
- EOV (non-VSAM).

The recovery procedures are similar for VSAM and non-VSAM operations.

Failures during SCRATCH or DELETE

System failures that occur when scratching (or deleting) RACF-protected DASD data sets can cause deletion of the data set profile in the RACF database even though the RACF indicator is left on.

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the data set profile for the affected data set.
3. If the volume information in the data set profile still exists, rerun the SCRATCH (or DELETE) operation.
4. If the volume information in the data set profile does not exist, use the TSO LISTDS (or access method services LISTCAT) command to determine if the RACF indicator is set in the DSCB (or catalog entry) for the data set.
5. If the RACF indicator is still set, enter the ALTDSD command with the NOSET and ADDVOL operands to re-create the information in the data set profile. If the data set profile does not exist, enter the ADDSD command with the NOSET operand to re-create it. Then rerun the SCRATCH (or DELETE) operation.

Failures during ALLOCATE or DEFINE

Failures that occur when allocating (or defining) RACF-protected DASD data sets can cause the data set profile to be created without setting the RACF indicator in the DSCB (or catalog entry).

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the data set profile.
3. If the profile does not exist, rerun the ALLOCATE (or DEFINE) operation.
4. If a profile exists, use the TSO LISTDS (or access method services LISTCAT) command to determine if the RACF indicator is set in the DSCB (or catalog entry) for the data set.
5. If the RACF indicator is not set, perform the following steps:
   a. Enter the DELDSD command with the NOSET operand to delete the data set profile.
   b. Rerun the ALLOCATE (or DEFINE) operation.

Failures during RENAME or ALTER

Failures that occur when renaming a RACF-protected DASD data set can cause discrepancies between the name in the data set profile and the name in the DSCB (or catalog entry).

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the data set profile.
3. If the data set profile has not been updated with the new name, rerun the RENAME (or ALTER) operation.
4. If the name in the data set profile has been updated but the name in the DSCB (or catalog entry) has not been updated, then perform the following steps:

a. Enter the ADDSD command with the NOSET operand and use the old data set name.
b. Enter the PERMIT command with the FROM operand using the new data set name.
c. Enter the DELDSD command with the NOSET operand and use the new data set name.
d. Rerun the RENAME (or ALTER) operation.

Failures during EOV (Non-VSAM)

Failures that occur during end-of-volume (EOV) processing can cause discrepancies between the DASD data set profile and the RACF indicator in the DSCB for that volume.

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. List the data set profile.
3. If no change has been made to the data set profile, rerun the step or job containing the EOV operation.
4. If the data set profile has been updated with the volume, use the TSO LISTDS command to determine if the RACF indicator is set in the DSCB for the volume.
5. If the RACF indicator for the volume is not set, perform the following steps:
   a. Enter the ALTDSD command with the NOSET and DELVOL operands.
   b. Rerun the step or job containing the EOV operation.

Failures in the RACF Subsystem Address Space

Failures can occur in the RACF subsystem address space for a number of reasons.

Recovering from RACF Parameter Library Problems

Several errors related to the RACF parameter library can occur. For all of these errors, RACF issues an error message and subsystem initialization completes, but no RRSF configuration occurs. Errors that can occur are:

- **Error:** The data set specified for the RACF parameter library in the RACFPARM DD statement does not exist.

  Recovery: Correct your RACFPARM DD statement to specify an existing data set, or create the data set you specified on the RACFPARM DD statement. The RACF subsystem address space must then be reinitialized to pick up the changes. To accomplish this, issue the RACF STOP command followed by the MVS START command with SUB=MSTR specified.
- **Error:** The EXEC statement in the JCL for the RACF procedure specifies a parameter library member on the PARM='OPT=*xx*' parameter, but no RACFPARM DD statement is included in the JCL.

  Recovery: Add the RACFPARM DD statement to your JCL. The RACF subsystem address space must then be reinitialized to pick up the changes. To accomplish this, issue the RACF STOP command followed by the MVS START command with SUB=MSTR specified.
- **Error:** The EXEC statement in the JCL for the RACF procedure specifies a parameter library member on the PARM='OPT=*xx*' parameter, but the parameter library member does not exist in the data set specified in the RACFPARM DD statement.

Recovery: If an appropriate parameter library member exists (for example if the PARM='OPT=*xx*' parameter is mistyped) you can issue a SET INCLUDE command to process the commands in that member. This is a temporary recovery method, and you will have to repeat it every time you reinitialize the RACF subsystem address space until you correct your JCL.

If no appropriate parameter library member exists, you can create one with the suffix you specified on the PARM='OPT=*xx*' parameter and issue a SET INCLUDE command to process the commands in the new member. The new member will be processed automatically whenever the RACF subsystem address space reinitializes in the future.

- **Error:** The EXEC statement in the JCL for the RACF procedure does not specify a parameter library member on the PARM='OPT=*xx*' parameter, and the JCL includes a RACFPARM DD statement but the data set it specifies does not include an IRROPT00 member.

  Recovery: This situation does not always occur as a result of an error. RACF assumes that if you include a RACFPARM DD statement in your JCL, you want a parameter library member processed automatically during initialization of the RACF subsystem address space. If you don't specify a member on the PARM='OPT=*xx*' parameter, RACF attempts to process member IRROPT00. However, you might want to use the RACF parameter library without having a member processed automatically. If this is the case, ignore the error message. Make sure that you do not create an IRROPT00 member, unless you want RACF to process it automatically every time the RACF subsystem address space reinitializes.

  If your intent was to have the IRROPT00 member automatically processed, create it. You can then issue a SET INCLUDE member to process the commands in the new member. The new member will be processed automatically whenever the RACF subsystem address space reinitializes in the future.

- **Error:** The JCL includes a RACFPARM DD statement, and the data set it specifies is empty. An abend message is issued, but the abend is not taken.

  Recovery: Make sure that the RACFPARM DD statement specifies the correct data set, and correct it if it does not. If it does, either add data to the data set, or remove the RACFPARM DD statement from the JCL. The RACF subsystem address space must then be reinitialized to pick up the changes. To accomplish this, issue the RACF STOP command followed by the MVS START command with SUB=MSTR specified.

Recovering When a Task Stops

When RACF detects that a task in the RACF subsystem address space has stopped, it tries to restart the task. RACF makes a number of attempts to restart the task, and if it is unsuccessful, gives up and issues error message IRRB041I. The error message identifies the module name that it cannot restart. When this happens, use the RESTART command to try to restart the task associated with the module. See "Restarting a function in the RACF subsystem" on page 76 for more information. If you cannot restart the task, use local procedures to determine what the problem is and correct it, and then restart the task.

It is possible that a task stops and RACF does not detect the problem, because it is busy doing something else. In this case, it will appear as if something that should be happening is not. For example, you might enter RACLINK commands but nothing happens. Or output from directed commands that you know executed might not be returning to the RRSFLIST data sets. If this is the case, use the RESTART command to restart the task associated with the actions that are not happening. For

example, if nothing happens when you enter RACLINK commands, try restarting the RACLINK task. If output is not returning to the RRSFLIST data sets, try restarting the OUTPUT task.

Recycling an RRSF Connection

Any of the sending or receiving device driver tasks can hang, requiring a recycle of the connection. There are several actions you can take to recycle a connection, listed below in the order of their power. However, as the power of each action increases, so does the possibility that requests will be lost.

1. The preferred way to recycle a connection is to use the TARGET command to make the connection dormant and then operative. However, this method will fail if a task is hung waiting for outstanding work. To restart the connection with node NEWYORK, enter:

   ```
   prefixTARGET NODE(NEWYORK) DORMANT
   prefixTARGET NODE(NEWYORK) OPERATIVE
   ```

2. If the TARGET command fails to fix the problem, use the RACF RESTART command to restart the connection with the node. The RESTART command restarts a connection even if a task is hung. To restart the connection with node NEWYORK, enter:

   ```
   prefixRESTART CONNECTION NODE(NEWYORK)
   ```

3. If that fails, use the RACF RESTART command to restart the connection task. Enter:

   ```
   prefixRESTART CONNECTION
   ```

4. As a last resort, stop and restart the RACF subsystem address space, using the RACF STOP and MVS START commands:

   ```
   prefixSTOP
   START subsystem_name SUB=MSTR
   ```

Recovering from VSAM Errors on the RRSF Workspace Data Sets

VSAM failures on the RRSF workspace data sets are critical. These data sets are used to checkpoint remote requests and the output returned from them, to maintain the integrity of the local and remote RACF databases. When a VSAM error occurs on a workspace data set that prevents RACF from writing records to or deleting records from the data set, RACF shuts down the connection, writes a message to the system console, creates a symptom record in SYS1.LOGREC, and attempts to close and deallocate the VSAM file that is experiencing the error.

Use local procedures to diagnose and correct the problem. After you have corrected the error, you must do the following to reactivate the connection:

```
TARGET NODE(nodename) DORMANT
TARGET NODE(nodename) WORKSPACE(workspace information)
TARGET NODE(nodename) OPERATIVE
```

Viewing the Workspace Data Sets

RACF provides a VSAM file browser utility (IRRBRW00) that transcribes workspace data set VSAM file records into a browsable output data set. It is provided in case off-line diagnosis of the RRSF workspace data set VSAM files is required. See the RACJCL member of SYS1.SAMPLIB for instructions on running the utility, and *z/OS Security Server RACF Diagnosis Guide* for information on setting up proper security to control its use.

Recovering When the Workspace Data Sets Fill Up

It is important to prevent the workspace data sets from filling up. If they do fill up, requests might be rejected and database inconsistencies might occur. The

procedure shown here can be used to increase the size of the workspace data sets before a problem occurs. The procedure can also be used to recover after the data sets have filled up.

Each workspace data set deals only with the connection between the local node and a single target node. During this procedure, any work originating at the local node to be sent to the remote node will be lost. Work which was previously saved in the workspace data sets will not be lost. Work originating at the remote node destined for the local node will not be lost.

In this example, assume that you want to increase the size of node ATLANTA's INMSG workspace data set used in communicating with node RALEIGH. The name of the workspace data set depends on the name of the RACF subsystem, the PREFIX information specified on the TARGET command, and the LU names of the local and remote nodes. You could determine the name of this workspace data set using the TARGET NODE(RALEIGH) LIST command. In this example, assume that the name of this workspace data set is RRSF.WORK.ATLLU.RALLU.INMSG.

1. On the local node, issue a TARGET command to make the connection with the remote node dormant. This insures that work destined for the local node from the remote node is queued at the remote node. For this example, from ATLANTA issue:

   ```
   TARGET NODE(RALEIGH) DORMANT
   ```

2. The workspace data sets are VSAM files. From TSO, create a new VSAM file using the IDCAMS DEFINE CLUSTER command. Use the MODEL parameter to insure that the new VSAM file has the same properties as the original VSAM file for the workspace data set. Use the RECORDS() parameter to override the insufficient size of the original VSAM file. Give the file a temporary name. For example:

   ```
   DEFINE CLUSTER(
     NAME('TEMP.INMSG')
     MODEL('RRSF.WORK.ATLLU.RALLU.INMSG')
     RECORDS(1000 750))
   ```

   SYS1.SAMPLIB member IRRSRRSF contains a sample member RRSFALOC with sample JCL to define the VSAM workspace data sets.

3. Issue a TARGET DELETE command to delete the remote node. This command causes RACF to deallocate the original VSAM file and release its control over the file. For example, from ATLANTA issue:

   ```
   TARGET NODE(RALEIGH) DELETE
   ```

4. Copy the original VSAM file into the new VSAM file, using the TSO REPRO command. For example:

   ```
   REPRO IDS('RRSF.WORK.ATLLU.RALLU.INMSG') ODS('TEMP.INMSG')
   ```

5. Delete the original VSAM file.

   ```
   DELETE 'RRSF.WORK.ATLLU.RALLU.INMSG'
   ```

6. Rename the new VSAM file to the name of the original VSAM file:

   ```
   ALTER 'TEMP.INMSG' NEWNAME('RRSF.WORK.ATLLU.RALLU.INMSG')
   ALTER 'TEMP.INMSG.*' NEWNAME('RRSF.WORK.ATLLU.RALLU.INMSG.*')
   ```

   An alternative is to create a new RRSF.WORK.ATLLU.RALLU.INMSG file, as shown in step 2, specifying TEMP.INMSG on the MODEL keyword, copy the information stored in TEMP.INMSG into it, and then delete TEMP.INMSG.

7. Update your RACF parameter library to reflect the new file size on TARGET commands. The new file size is the first number on the RECORDS keyword in

the DEFINE CLUSTER command, shown in step 2 on page 330. For example, if a member of the RACF parameter library on ATLANTA contains the following command:

```
TARGET NODE(RALEIGH) PREFIX(RRSF.WORK) FILESIZE(500)
```

change it to:

```
TARGET NODE(RALEIGH) PREFIX(RRSF.WORK) FILESIZE(1000)
```

If you fail to update the RACF parameter library, and the node is ever deleted while the VSAM file is empty, RACF deletes the VSAM file. Then, the next time the parameter library is processed, RACF will re-create the file using the smaller file size.

8. Issue the TARGET command to redefine the connection to the remote node, specifying the appropriate configuration, protocol, and workspace information. (Depending on how you have set up your RACF parameter library, you might have already defined a parameter library member containing a TARGET command that does this, and updated the TARGET command in the preceding step. If so, you can issue a SET INCLUDE command to execute that member.)

Notes:

1. After you perform this procedure, the INMSG file for the RALEIGH node might be a different size than the OUTMSG file, and could be on a different volume. If you do a TARGET NODE(RALEIGH) LIST command, the output will reflect what was entered on the TARGET command that established the connection, and this might not be the values currently in effect.
2. If an INMSG file becomes full, RACF deallocates it, yet might still process the work (based on in-storage queues). However, when each piece of work is done, RACF cannot delete the INMSG file record for it while the INMSG file is deallocated. Consequently, if the INMSG file, or a copy of it, is made operative again, the work might be executed again.

The Last Resort—Shutting Down the RACF Subsystem Address Space

When other recovery methods fail to fix RACF subsystem address space failures, as a last resort try shutting down and restarting the RACF subsystem address space, using the RACF STOP and MVS START commands:

```
prefixSTOP
START subsystem_name,SUB=MSTR
```

See “Stopping the RACF subsystem address space” on page 78 and “Restarting the RACF subsystem” on page 76 for more information.

Chapter 10. Storage Estimates

RACF Database Storage Requirements 333
Factors Affecting the Size of the RACF Database. 333
Formula for the RACF Database Size 333
RACF Virtual Storage Requirements 336
Coupling Facility Cache Structure Storage Requirements 338

This chapter provides information for estimating RACF storage requirements.

RACF Database Storage Requirements

This section explains how to estimate the size of a RACF database. Note that when a RACF database becomes full, you can extend it with the Split/Merge/Extend utility program (IRRUT400). You can determine how full your database is with the database verification utility program (IRRUT200).

Factors Affecting the Size of the RACF Database

The direct access space needed for a RACF database depends mainly on the number of users, groups, user-group connections, and resource profiles defined to RACF. Other factors that might affect the amount of space required for the database include the following:

- The length of the names of the entities defined to RACF
- Activating the RACGLIST class
- Activating the CACHECLS class
- How efficiently the space in the RACF database is utilized

Formula for the RACF Database Size

You can use the formula in Table 16 to estimate the size required for a RACF database.

Table 16. Formula for the RACF Database Size

The number of 4K (4096) blocks required = $12 + A + B + B_A + C$ where: 12 = the number of blocks required for the ICB and database templates.

© Copyright IBM Corp. 1994, 2004

Table 16. Formula for the RACF Database Size (continued)

```
A  = the number of blocks required for the profiles:

A = F + G + H + I + J + K + L + M + N + O + P + Q + R + S + T + U + V + W + X + Y + Z
   --- --- --  --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
    16  16  16  16  16  16  16  3   16  16  16  2   16  16  8   16  16  16   16  16  8

    + AA + BB + CC + DD + EE + FF + GG + HH + II + JJ + KK + LL + MM + NN + OO
     ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---
      16   16   16   16   16    3   16    3   16    2   16   16   16   16   16

where:
    F = the number of users defined to RACF
    G = the number of groups defined to RACF
    H = the number of data sets defined to RACF
    I = the number of general resources defined to RACF
    J = the number of user
```

CICS segments K = the number of user DCE segments L = the number of user DFP segments M = the number of user EIM segments N = the number of user KERB segments O = the number of user LANGUAGE segments P = the number of user LNOTES segments * Q = the number of user PROXY segments R = the number of user NDS segments * S = the number of user NETVIEW segments T = the number of user OMVS segments * U = the number of user OPERPARM segments V = the number of user OVM segments W = the number of user TSO segments X = the number of user WORKATTR segments Y = the number of group DFP segments Z = the number of group OMVS segments * AA = the number of group OVM segments BB = the number of group TME segments CC = the number of data set DFP segments DD = the number of data set TME segments EE = the number of general resource ALIAS segments FF = the number of general resource CERTDATA segments GG = the number of general resource DLFDATA segments HH = the number of general resource EIM segments II = the number of general resource KERB segments JJ = the number of general resource PROXY segments KK = the number of general resource SESSION segments LL = the number of general resource STDATA segments MM = the number of general resource SSIGNON segments NN = the number of general resource SVFMR segments OO = the number of general resource TME segments

Notes:

1. The divisor of 16 is used in most cases because, in most cases, approximately 16 profiles (or segments) fit in one 4096-byte block. However, the profile (or segment) size is variable depending upon installation data, the number of users in a group, the number of entries on a data set or resource access list, and so on. You might want to replace the divisor 16 in one or more factors with 15 or less if the profiles (or segments) are longer than 256 bytes on the average. Because digital certificates are often 1K to 2K bytes in size, a divisor of 3 is used for the CERTDATA segments. Because OMVS segments are large, a divisor of 8 is used for the user and group OMVS segments. A divisor of 2 is used for PROXY segments, and 3 for EIM segments, because these segments are also large.
2. The values for P, R, T, and Z should be set to zero if your database is using stage 3 of application identity mapping. At this stage, RACF no longer maintains mapping profiles that correspond to alias index entries in the LNOTES, NDS, or OMVS segments.

Table 16. Formula for the RACF Database Size (continued)

B = the number of index blocks required:

B = sum from I=1 to 10 of $D/(E^i)$

D = F + G + H + I + J + K + L + M + N + O + P + Q + R + S + T + U + V + W + X + Y + Z
+ AA + BB + CC + DD + EE + FF + GG + HH + II + JJ + KK+ LL+ MM + NN + OO

E = the number of names that fit into an index block, which is approximately:

$$\frac{4096 \times .5}{10 + \text{average name length}}$$

Notes:

1. The formula for B is equivalent to:
   $B = D/E^1 + D/E^2 + \ldots + D/E^9 + D/E^{10}$
2. The formula for E assumes that the index blocks are half full (.5). If you can extend an existing RACF data set with the IRRUT400 utility, then you can determine (with the IRRUT200 utility program) how full the index blocks are on the existing data set and replace the .5 value with a value that you determine. In this case, you can also consider the compressed name length when specifying the average index name length.

B_A = the number of blocks required for the alias index
(zero if running application identity mapping in stage=0)

$B_A \sim$ the number of blocks required for alias index sequence set, which is a good approximation because the space requirement for higher-level index entries is a small fraction of this number

$$B_A \sim \frac{D_A}{E_A}$$

where:

$D_A = P + R + T + Z + EE$

$$E_A = \frac{4096 \times 0.5}{18 + \text{average user or group name length} + 3 + \text{average alias name length}}$$

18 = the length of the alias index entry overhead

3 = the length of the alias index entry key prefix

Note: The calculation for E_A assumes the following:

1. The alias index blocks are half full (0.5). If you can extend an existing RACF data set with the IRRUT400 utility, you can:
   - Use the IRRUT200 utility program to determine how full the alias index blocks are on the existing data set
   - Replace the 0.5 value with a value that you determine

   In this case, you can also consider the compressed name length when specifying the average alias name length.
2. Each alias entry maps to a single user or group profile. While it is possible for multiple users to share a single UID or multiple groups to share a single GID, the implementation is not recommended.
3. The average alias name length is the average length of your SNAME, UNAME, GID, UID and IPLOOK values.

C = the number of BAM blocks required. One BAM block is required for every 2038 blocks in the

RACF data set. C = 13 + A + B ---------- where A and B are as described above 2038

Note: Round C up to a whole number.

RACF Virtual Storage Requirements

Figure 34 is a storage map for RACF. Table 17 on page 337 gives virtual storage requirements for RACF.

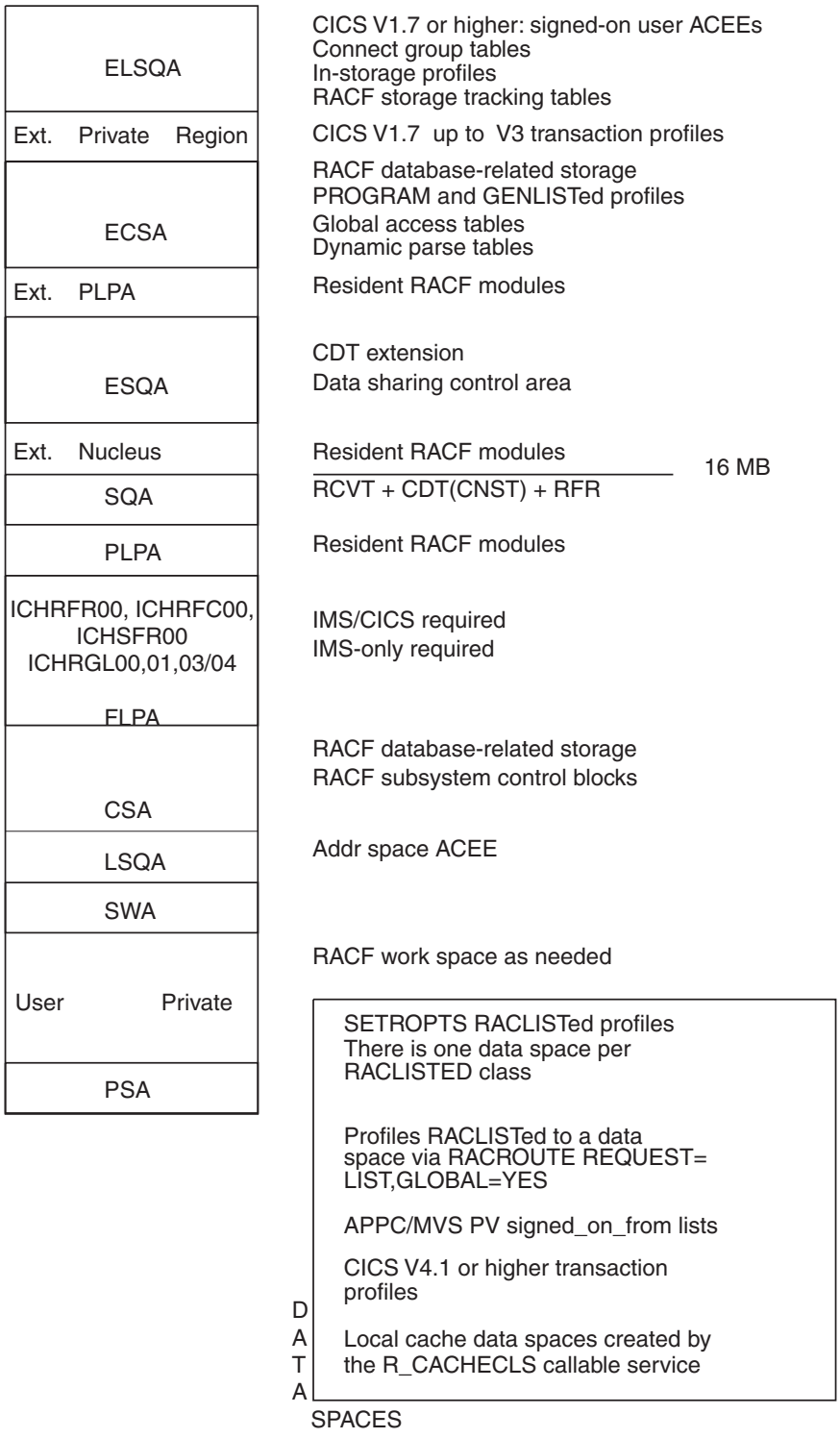


Figure 34. RACF Storage Use

Table 17. RACF Estimated Storage Usage

Storage Subpool	Usage	How to Estimate Size
FLPA	RACF service routines, if IMS or CICS is using RACF for authorization checking	47 000
	RACROUTE REQUEST=FASTAUTH and ICHRTX00 exits	Measure using AMBLIST
PLPA	RACF installation exits that are AMODE(24) or AMODE(ANY)	Measure using AMBLIST
	RACF RMODE(24) code	750
	RACF service routines, if IMS or CICS is not using RACF for authorization checking, unless explicitly removed from SYS1.LPALIB and placed elsewhere for use in FLPA	47 000
	RACROUTE REQUEST=FASTAUTH and ICHRTX00 exits	Measure using AMBLIST
	RACF range table	4 + (number_of_ranges × 45)
EPLPA	RACF installation exits that are AMODE(31)	Measure using AMBLIST
	RACF resident modules above 16MB	875 000
SQA	RACF communications vector table and extension	2800
	Class descriptor table (CNST) and RACF router table	7500 + 58 × number_of_installation-defined_classes
ESQA	RACF data sharing control area	300 (when enabled for sysplex communication)
	Class descriptor table (CNSX)	(number_of_classes_IBM_supplies × 28) + (number_of_router_table_entries_IBM_supplies × 30) + (number_of_installation-defined_classes × 58) + 26 IBM supplies 196 classes and 215 router table entries, so the size of the CNSX is 11 938 + (number_of_installation-defined_classes × 58). If you install a PTF that adds entries, you will need to recalculate this number.
LSQA	ACEE and related storage **Notes:** 1. Applications can place this storage in a different subpool. 2. Applications can create multiple ACEEs in this and other storage subpools.	400 + installation_data_length + terminal_installation_data_length + application_installation_data + (52 for every 78 temporary data sets, rounded up to the next multiple of 52) If the address space has been dubbed an OS/390 UNIX process, then add: 52 + (number_of_connected_groups_with_GIDs × 4) Add 112 bytes if the user has CLAUTH for a class with a POSIT value over 127.
ELSQA	Connect group table	64 + (48 × number_of_groups_connected)
	In-storage generic profiles	160 + number_of_generic_profiles × (14 + average_profile_size + average_profile_name_length)
	RACF storage tracking table	3500
	RACROUTE REQUEST=LIST profiles **Note:** Applications can place these profiles in a different storage subpool.	2108 + (number_of_profiles_in_class × 16) + (number_of_unique_generic_profile_prefix_lengths × 24) + (number_of_generic_profiles × 4) + (number_of_resident_profiles × (10 + average_profile_size + (1.5 × class_max_profile_name_size))) for each class if GLOBAL=YES is not specified
CSA	RACF database control structures (DCB, DEB, templates)	4600 + (number_of_BAM_blocks × 6) + (364 x number_of_RACF_primary_data_sets)
	RACF subsystem control blocks	3500

Table 17. RACF Estimated Storage Usage (continued)

Storage Subpool	Usage	How to Estimate Size
ECSA	RACF data set descriptor table and extension	168 + (896 × number_of_RACF_primary_data_sets)
	RACF ICB (non-shared DB)	4096 per RACF database if the database is not shared and is not on a device marked as shared, 0 otherwise
	RACF global access tables	3040 + (number_of_user_classes × 24) + 2 × (18 + number_of_entries × (6 + (1.5 × max_profile_name_size)))
	RACF program control table	28 + (number_of_program_profiles × average_program_profile_size) + (number_of_controlled_libraries × 50) To find the average_program_profile_size, use the following formula: 54 + (average_number_of_access _entries × 9) + (average_number_of _conditional_access_entries × 17) + (average_number_of_libraries × 52)
	RACF resident data blocks	For each primary data set: 3248 + (4136 × number_of_database_buffers) If using sysplex communication, for each backup data set add: 3248 + (4136 × number_of_database_buffers × 2)
	Dynamic parse tables	30 000
	SETROPTS GENLIST profiles	52 + (number_of_profiles_in_class × 16) + (number_of_resident_profiles × (10 + average_profile_size + (1.5 × class_max_profile_name_size)))
	Alias-related template extension	1296
User private Below 16MB	RACF transient storage	16 000 (minimum) while a RACF service is executing

Coupling Facility Cache Structure Storage Requirements

See “Defining RACF structures for the coupling facility” on page 90 for information on how to calculate the storage required for coupling facility cache structures.

Appendix A. Supplied CDT classes

This appendix describes the general resource classes you can find in the supplied class descriptor table (CDT) and contains the following sections:

- "Supplied resource classes for z/OS and OS/390 systems"
- "Supplied resource classes for z/VM and VM systems" on page 346

See *z/OS Security Server RACF Macros and Interfaces* to find details (such as POSIT values) for each class.

Supplied resource classes for z/OS and OS/390 systems

Table 18 lists the supplied CDT classes that can be used on z/OS and OS/390 systems.

Table 18. Resource Classes for z/OS and OS/390 Systems

Class Name	Description
ALCSAUTH	Supports the Airline Control System/MVS (ALCS/MVS) product.
APPCLU	Verifying the identity of partner logical units during VTAM session establishment.
APPCPORT	Controlling which user IDs can access the system from a given LU (APPC port of entry). Also, conditional access to resources for users entering the system from a given LU.
APPCSERV	Controlling whether a program being run by a user can act as a server for a specific APPC transaction program (TP).
APPCSI	Controlling access to APPC side information files.
APPCTP	Controlling the use of APPC transaction programs.
APPL	Controlling access to applications.
CACHECLS	Contains profiles used for saving and restoring cache contents from the RACF database.
CBIND	Controlling the client's ability to bind to the server.
CONSOLE	Controlling access to MCS consoles. Also, conditional access to other resources for commands originating from an MCS console.
CSFKEYS	Controlling use of Integrated Cryptographic Service Facility (ICSF) cryptographic keys. See also the GCSFKEYS class.
CSFSERV	Controlling use of Integrated Cryptographic Service Facility (ICSF) cryptographic services.
DASDVOL	DASD volumes. See also the GDASDVOL class.
DBNFORM	Reserved for future IBM use.
DEVICES	Used by MVS allocation to control who can allocate devices such as: • Unit record devices (printers and punches) (allocated only by PSF, JES2, or JES3) • Graphics devices (allocated only by VTAM) • Teleprocessing (TP) or communications devices (allocated only by VTAM)
DIGTCERT	Contains digital certificates and information related to them.
DIGTCRIT	Specifies additional criteria for certificate name filters.
DIGTNMAP	Mapping class for certificate name filters.

© Copyright IBM Corp. 1994, 2004

Table 18. Resource Classes for z/OS and OS/390 Systems (continued)

Class Name	Description
DIGTRING	Contains a profile for each key ring and provides information about the digital certificates that are part of each key ring.
DIRAUTH	Setting logging options for RACROUTE REQUEST=DIRAUTH requests. Also, if the DIRAUTH class is active, security label authorization checking is done when a user receives a message sent through the TPUT macro or the TSO SEND, or LISTBC commands. Profiles are not allowed in this class.
DLFCLASS	The data lookaside facility.
FACILITY	Miscellaneous uses. Profiles are defined in this class so that resource managers (typically program products or components of MVS or VM) can check a user's access to the profiles when the users take some action. Examples are catalog operations (DFP) and use of the vector facility (an MVS component). RACF does not document all of the resources used in the FACILITY class by other products. For information on the FACILITY class resources used by a specific product (other than RACF itself), see the product's documentation.
FIELD	Fields in RACF profiles (field-level access checking).
GCSFKEYS	Resource group class for CSFKEYS class. [1]
GDASDVOL	Resource group class for DASDVOL class. [1]
GLOBAL	Global access checking table entry. [1]
GMBR	Member class for GLOBAL class (not for use on RACF commands).
GSDSF	Resource group class for SDSF class. [1]
GTERMINL	Resource group class for TERMINAL class. [1]
IBMOPC	Controlling access to OPC/ESA subsystems.
JESINPUT	Conditional access support for commands or jobs entered into the system through a JES input device.
JESJOBS	Controlling the submission and cancellation of jobs by job name.
JESSPOOL	Controlling access to job data sets on the JES spool (that is, SYSIN and SYSOUT data sets).
KEYSMSTR	Contains profiles that hold keys to encrypt data stored in the RACF database, such as LDAP BIND passwords and DCE passwords.
LDAPBIND	Contains the LDAP server URL, bind distinguished name, and bind password.
LOGSTRM	Controls system logger resources, such as log streams and the coupling facility structures associated with log streams.
NODES	Controlling the following on MVS systems: • Whether jobs are allowed to enter the system from other nodes • Whether jobs that enter the system from other nodes have to pass user identification and password verification checks
NODMBR	Member class for NODES class (not for use on RACF commands).
OPERCMDS	Controlling who can issue operator commands (for example, JES and MVS, and operator commands). [2]
PMBR	Member class for PROGRAM class (not for use on RACF commands).
PROGRAM	Controlled programs (load modules). [1]

Table 18. Resource Classes for z/OS and OS/390 Systems (continued)

Class Name	Description
PROPCNTL	Controlling if user ID propagation can occur, and if so, for which user IDs (such as the CICS or IMS main task user ID), user ID propagation is *not* to occur.
PSFMPL	Used by PSF to perform security functions for printing, such as separator page labeling, data page labeling, and enforcement of the user printable area.
PTKTDATA	PassTicket key class enables the security administrator to associate a RACF secured signon secret key with a particular mainframe application that uses RACF for user authentication. Examples of such applications are IMS, CICS, TSO, VM, APPC, and MVS batch.
RACGLIST	Class of profiles that hold the results of RACROUTE REQUEST=LIST,GLOBAL=YES or a SETROPTS RACLIST operation.
RACFVARS	RACF variables. In this class, profile names, which start with & (ampersand), act as RACF variables that can be specified in profile names in other RACF general resource classes.
RRSFDATA	Used to control RACF remote sharing facility functions.
RVARSMBR	Member class for RACFVARS (not for use on RACF commands).
SCDMBR	Member class for SECDATA class (not for use on RACF commands).
SDSF	Controls the use of authorized commands in the System Display and Search Facility (SDSF). See also GSDSF class.
SECDATA	Security classification of users and data (security levels and security categories). [1]
SECLABEL	If security labels are used, and, if so, their definitions. [2]
SECLMBR	Member class for the SECLABEL class (not for use on RACF commands).
SERVAUTH	Contains profiles used by servers to check a client's authorization to use the server or to use resources managed by the server. Also, can be used to provide conditional access to resources for users entering the system from a given server.
SERVER	Controlling the server's ability to register with the daemon.
SMESSAGE	Controlling to which users a user can send messages (TSO only).
SOMDOBJS	Controlling the client's ability to invoke the method in the class.
STARTED	Used in preference to the started procedures table to assign an identity during the processing of an MVS START command.
SURROGAT	If surrogate submission is allowed, and if allowed, which user IDs can act as surrogates.
SYSMVIEW	Controlling access by the SystemView® for MVS Launch Window to SystemView for MVS applications.
TAPEVOL	Tape volumes.
TEMPDSN	Controlling who can access residual temporary data sets. You cannot create profiles in this resource class.
TERMINAL	Terminals (TSO or VM). See also GTERMINL class.
VTAMAPPL	Controlling who can open ACBs from non-APF authorized programs.
WRITER	Controlling the use of JES writers.
	CICS classes
ACICSPCT	CICS program control table. [2]

Table 18. Resource Classes for z/OS and OS/390 Systems (continued)

Class Name	Description
BCICSPCT	Resource group class for ACICSPCT class. [1]
CCICSCMD	Used by CICS/ESA 3.1, or later, to verify that a user is permitted to use CICS system programmer commands such as INQUIRE, SET, PERFORM, and COLLECT. [1]
CPSMOBJ	Used by CICSPlex® System Manager, which provides a central point of control when running multiple CICS systems, to determine operational controls within a CICS complex.
CPSMXMP	Used by CICSPlex System Manager to identify exemptions from security controls within a CICS complex.
DCICSDCT	CICS destination control table. [2]
ECICSDCT	Resource group class for DCICSDCT class. [1]
FCICSFCT	CICS file control table. [2]
GCICSTRN	Resource group class for TCICSTRN class. [2]
GCPSMOBJ	Resource grouping class for CPSMOBJ.
HCICSFCT	Resource group class for FCICSFCT class. [1]
JCICSJCT	CICS journal control table. [2]
KCICSJCT	Resource group class for JCICSJCT class. [1]
MCICSPPT	CICS processing program table. [2]
NCICSPPT	Resource group class for MCICSPPT class. [1]
PCICSPSB	CICS program specification blocks or PSBs
QCICSPSB	Resource group class for PCICSPSB class. [1]
SCICSTST	CICS temporary storage table. [2]
TCICSTRN	CICS transactions.
UCICSTST	Resource group class for SCICSTST class. [1]
VCICSCMD	Resource group class for the CCICSCMD class. [1]
	DB2 classes
DSNADM	DB2 administrative authority class.
DSNR	Controls access to DB2 subsystems.
GDSNBP	Grouping class for DB2 buffer pool privileges.
GDSNCL	Grouping class for DB2 collection privileges.
GDSNDB	Grouping class for DB2 database privileges.
GDSNJR	Grouping class for Java archive files (JARs).
GDSNPK	Grouping class for DB2 package privileges.
GDSNPN	Grouping class for DB2 plan privileges.
GDSNSC	Grouping class for DB2 schemas privileges.
GDSNSG	Grouping class for DB2 storage group privileges.
GDSNSM	Grouping class for DB2 system privileges.
GDSNSP	Grouping class for DB2 stored procedure privileges.
GDSNSQ	Grouping class for DB2 sequences.
GDSNTB	Grouping class for DB2 table, index, or view privileges.
GDSNTS	Grouping class for DB2 tablespace privileges.

Table 18. Resource Classes for z/OS and OS/390 Systems (continued)

Class Name	Description
GDSNUF	Grouping class for DB2 user-defined function privileges.
GDSNUT	Grouping class for DB2 user-defined distinct type privileges.
MDSNBP	Member class for DB2 buffer pool privileges.
MDSNCL	Member class for DB2 collection privileges.
MDSNDB	Member class for DB2 database privileges.
MDSNJR	Member class for Java archive files (JARs).
MDSNPK	Member class for DB2 package privileges.
MDSNPN	Member class for DB2 plan privileges.
MDSNSC	Member class for DB2 schema privileges.
MDSNSG	Member class for DB2 storage group privileges.
MDSNSM	Member class for DB2 system privileges.
MDSNSP	Member class for DB2 stored procedure privileges.
MDSNSQ	Member class for DB2 sequences.
MDSNTB	Member class for DB2 table, index, or view privileges.
MDSNTS	Member class for DB2 tablespace privileges.
MDSNUF	Member class for DB2 user-defined function privileges.
MDSNUT	Member class for DB2 user-defined distinct type privileges.
	DCE classes
DCEUUIDS	Used to define the mapping between a user's RACF user ID and the corresponding DCE principal UUID.
	Enterprise Java Beans classes
EJBROLE	Member class for Enterprise Java Beans authorization roles.
GEJBROLE	Grouping class for Enterprise Java Beans authorization roles.
JAVA	Contains profiles that are used by Java for z/OS applications to perform authorization checking for Java for z/OS resources.
	IMS classes
AIMS	Application group names (AGN).
CIMS	Command.
DIMS	Grouping class for command.
FIMS	Field (in data segment).
GIMS	Grouping class for transaction.
HIMS	Grouping class for field.
OIMS	Other.
PIMS	Database.
QIMS	Grouping class for database.
SIMS	Segment (in database).
TIMS	Transaction (trancode).
UIMS	Grouping class for segment.
WIMS	Grouping class for other.
	Infoprint® Server class

Table 18. Resource Classes for z/OS and OS/390 Systems (continued)

Class Name	Description
PRINTSRV	Controls access to printer definitions for Infoprint Server.
Information/Management (Tivoli Service Desk) classes	
GINFOMAN	Grouping class for Information/Management (Tivoli Service Desk) resources.
INFOMAN	Member class for Information/Management (Tivoli Service Desk) resources.
LFS/ESA classes	
LFSCLASS	Controls access to file services provided by LFS/ESA.
License Manager class	
ILMADMIN	Controls access to the administrative functions of IBM License Manager.
Lotus Notes for z/OS and Novell Directory Services for OS/390 classes	
NDSLINK	Mapping class for Novell Directory Services for OS/390 user identities.
NOTELINK	Mapping class for Lotus Notes for z/OS user identities.
MQSeries® classes	
GMQADMIN	Grouping class for MQSeries administrative options. [1]
GMQCHAN	Reserved for MQSeries.
GMQNLIST	Grouping class for MQSeries namelists. [1]
GMQPROC	Grouping class for MQSeries processes. [1]
GMQQUEUE	Grouping class for MQSeries queues. [1]
MQADMIN	Protects MQSeries administrative options.
MQCHAN	Reserved for MQSeries.
MQCMDS	Protects MQSeries commands.
MQCONN	Protects MQSeries connections.
MQNLIST	Protects MQSeries namelists.
MQPROC	Protects MQSeries processes.
MQQUEUE	Protects MQSeries queues.
NetView classes	
NETCMDS	Controlling which NetView commands the NetView operator can issue.
NETSPAN	Controlling which NetView commands the NetView operator can issue against the resources in this span.
NVASAPDT	NetView/Access Services.
PTKTVAL	Used by NetView/Access Services Secured Single Signon to store information needed when generating a PassTicket.
RMTOPS	NetView Remote Operations.
RODMMGR	NetView Resource Object Data Manager (RODM).
Network Authentication Service classes	
KERBLINK	Mapping class for user identities of local and foreign principals. [3]
REALM	Used to define the local and foreign realms. [3]
SMS (DFSMSdfp) classes	
MGMTCLAS	SMS management classes.
STORCLAS	SMS storage classes.

Table 18. Resource Classes for z/OS and OS/390 Systems (continued)

Class Name	Description
SUBSYSNM	Authorizes a subsystem (such as a particular instance of CICS) to open a VSAM ACB and use VSAM Record Level Sharing (RLS) functions.
	Tivoli classes
ROLE	Specifies the complete list of resources and associated access levels that are required to perform the particular job function this role represents and defines which RACF groups are associated with this role.
TMEADMIN	Maps the user IDs of Tivoli administrators to RACF user IDs.
	TSO classes
ACCTNUM	TSO account numbers.
PERFGRP	TSO performance groups.
TSOAUTH	TSO user authorities such as OPER and MOUNT.
TSOPROC	TSO logon procedures.
	z/OS UNIX classes
DIRACC	Controls auditing (using SETROPTS LOGOPTIONS) for access checks for read/write access to HFS directories. Profiles are not allowed in this class.
DIRSRCH	Controls auditing (using SETROPTS LOGOPTIONS) of HFS directory searches. Profiles are not allowed in this class.
FSOBJ	Controls auditing (using SETROPTS LOGOPTIONS) for all access checks for HFS objects except directory searches. Controls auditing (using SETROPTS AUDIT) of creation and deletion of HFS objects. Profiles are not allowed in this class.
FSSEC	Controls auditing (using SETROPTS LOGOPTIONS) for changes to the security data (FSP) for HFS objects. Profiles are not allowed in this class.
IPCOBJ	Controlling auditing and logging of IPC security checks.
PROCACT	Controls auditing (using SETROPTS LOGOPTIONS) of functions that look at data from, or affect the processing of, z/OS UNIX processes. Profiles are not allowed in this class.
PROCESS	Controls auditing (using SETROPTS LOGOPTIONS) of changes to UIDs and GIDs of z/OS UNIX processes. Controls auditing (using SETROPTS AUDIT) of dubbing and undubbing of z/OS UNIX processes. Profiles are not allowed in this class.
UNIXMAP	Contains profiles that are used to map z/OS UNIX UIDs to RACF user IDs and z/OS UNIX GIDs to RACF group names.
UNIXPRIV	Contains profiles that are used to grant z/OS UNIX privileges.

Notes:

1. You cannot specify this class name on the GENCMD, GENERIC, and GLOBAL/NOGLOBAL operands of the SETROPTS command.
2. You cannot specify this class name on the GLOBAL operand of the SETROPTS command or, if you do, the GLOBAL checking is not performed.
3. You cannot specify this class name on the GENCMD and GENERIC operands of the SETROPTS command.

Supplied resource classes for z/VM and VM systems

Table 19 lists the supplied classes you can use on z/VM and VM systems. These classes are primarily relevant if you share your RACF database with a z/VM or VM system.

Table 19. Resource Classes for z/VM and VM Systems

Class Name	Description
DIRECTRY	Protection of shared file system (SFS) directories.
FACILITY	Miscellaneous uses. Profiles are defined in this class so resource managers (typically program products or components of MVS or VM) can check a user's access to the profiles when the users take some action. Examples are using combinations of options for tape mounts, and use of the RACROUTE interface. RACF does not document all of the resources used in the FACILITY class by other products. For information on the FACILITY class resources used by a specific product (other than RACF itself), see that product's documentation.
FIELD	Fields in RACF profiles (field-level access checking).
FILE	Protection of shared file system (SFS) files.
GLOBAL	Global access checking. [1]
GMBR	Member class for GLOBAL class (not for use on RACF commands).
GTERMINL	Terminals whose IDs do not fit into generic profile naming conventions. [1]
PSFMPL	When class is active, PSF/VM performs separator and data page labeling as well as auditing.
PTKTDATA	PassTicket key class.
PTKTVAL	Used by NetView/Access Services Secured Single Signon to store information needed when generating a PassTicket.
RACFVARS	RACF variables. In this class, profile names, which start with & (ampersand), act as RACF variables that can be specified in profile names in other RACF general resource classes.
RVARSMBR	Member class for RACFVARS (not for use on RACF commands).
SCDMBR	Member class for SECDATA class (not for use on RACF commands).
SECDATA	Security classification of users and data (security levels and security categories). [1]
SECLABEL	If security labels are used and, if so, their definitions. [2]
SFSCMD	Controls the use of shared file system (SFS) administrator and operator commands.
TAPEVOL	Tape volumes.
TERMINAL	Terminals (TSO or VM). See also GTERMINL class.
VMBATCH	Alternate user IDs.
VMBR	Member class for VMEVENT class (not for use on RACF commands).
VMCMD	Certain CP commands and other requests on VM.
VMEVENT	Auditing and controlling security-related events (called VM events) on VM systems.
VMMAC	Used in conjunction with the SECLABEL class to provide security label authorization for some VM events. Profiles are not allowed in this class.
VMMDISK	VM minidisks.

Table 19. Resource Classes for z/VM and VM Systems (continued)

Class Name	Description
VMNODE	RSCS nodes.
VMRDR	VM unit record devices (virtual reader, virtual printer, and virtual punch).
VMSEGMT	Restricted segments, which can be named saved segments (NSS) and discontiguous saved segments (DCSS).
VXMBR	Member class for VMXEVENT class (not for use on RACF commands).
VMXEVENT	Auditing and controlling security-related events (called VM events) on VM systems.
VMPOSIX	Contains profiles used by OpenExtensions VM.
WRITER	VM print devices.

Notes:

1. You cannot specify this class name on the GENCMD, GENERIC, and GLOBAL/NOGLOBAL operands of the SETROPTS command.
2. You cannot specify this class name on the GLOBAL operand of the SETROPTS command or, if you do, the GLOBAL checking is not performed.

Appendix B. RRSF Initialization Worksheet and Scenario

In order to configure your RRSF network, you need to plan for the type of network you desire. This planning phase requires that you make certain decisions regarding the nodes you plan to have participate in the network. After you decide which nodes will participate in the network, the following worksheet will help you gather the appropriate information for the RACF parameter library and to build the desired network. The worksheet has been designed for you to make copies of it. It is suggested that you use those copies to mark your installation-specific information regarding the RRSF network you wish to configure.

You will need to complete one RRSF Node Configuration Worksheet for each single-system node in your RRSF network, and one for each system in each multisystem node in your RRSF network. At the bottom of the node's worksheet, you can identify which remote nodes will be enabled to communicate with this node and whether the connection will be operative or dormant. You will need to retrieve the necessary information from the remote node's RRSF Node Configuration Worksheet to complete the TARGET commands for each remote node.

A task-oriented scenario also follows to help reinforce the steps necessary to properly configure your network.

© Copyright IBM Corp. 1994, 2004

RRSF Node Configuration Worksheet

Local Node:

Name: _____

Type of Node:

____ Single-system ____ Multisystem

System Name: _____ Main: ___ Yes ___ No

Local node status:

____ Operative ____ Dormant

APPC protocol information:

LUNAME: _____ (from SYS1.PARMLIB(APPCPM*xx*))
TPNAME: _____ (default=IRRRACF)
MODENAME: _____ (default=IRRMODE)

Workspace information:

Prefix name for high level qualifier for data set names: _____
Filesize for workspace: _____ (initial=500)
____ **DFP SMS**
Name of SMS storage class: _____
Name of SMS data class: _____
Name of SMS management class: _____
____ **DFP Non-SMS**
Volume serial number to contain the workspace data sets: _____

RACF parameter library information:

Name of

RACF parameter library data set:

_____

Name of the member that is invoked automatically when the RACF subsystem initializes:
IRROPT_____ (default=IRROPT00)

Password synchronization:

____ Yes ____ No

Command direction:

____ Yes ____ No

Automatic direction:

____ Yes ____ No
Output Level: _____ (ALWAYS | WARN | FAIL | NOOUTPUT)
Notify Level: _____ (ALWAYS | WARN | FAIL | NONOTIFY)

Node and user ID	Receive Output?	Notification?
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

JESNODE: (for transmits)

_____ (to override the value obtained automatically)

Remote NODE/SYSNAME:

Status of the local node's connection with each remote node:

_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main

RRSF Initialization Scenario

Background Information

You are the system programmer responsible for customizing RACF. Systems SYSMVS01 and SYSMVS02 are running with the RACF subsystem active, and you want to take advantage of the RACF remote sharing facility (RRSF) by configuring SYSMVS01 and SYSMVS02 in a two-node configuration. You have the following information:

- System SYSMVS01:
  - You want the node name to be MVS01.
  - The system does not share its RACF database with other systems, so it will be configured as a single-system node.
  - The status of local node MVS01 is to be operative.
  - You want the connection with MVS02 to be operative.
  - The name of the RACF parameter library data set will be RRSF.PARM.
  - You want the member IRROPT01 in the RACF parameter library invoked automatically when the RACF subsystem initializes.
  - You have DFP Non-SMS.
  - The volume that will contain the RACF workspace data sets will be DASD01.
  - The high level qualifier for the workspace data sets will be SYS1.RACF.
  - The LUNAME is MF1AP001.
  - You already have the following JCL to activate the RACF subsystem:

    ```
    //RACF PROC
    //RRSF EXEC PGM=IRRSSM00
    ```

  - You want to activate automatic direction for the node and you want the OUTPUT level of FAIL and the NOTIFY level of FAIL for the user IDs: SECADM and SYSPRG on MVS01 and SECADM on MVS02.
  - APPC and VTAM have already been installed and configured.
  - The JESNODE that will be used for transmits will be THISJES.
- System SYSMVS02:
  - You want the node name to be MVS02.
  - The system does not share its RACF database with other systems, so it will be configured as a single-system node.
  - The status of local node MVS02 is to be operative.
  - You want the connection with MVS01 to be operative.
  - The name of the RACF parameter library data set will be RRSF.PARM.
  - You want the member IRROPT02 in the RACF parameter library invoked automatically when the RACF subsystem initializes.
  - You have DFP Non-SMS.
  - The volume that will contain the RACF workspace data sets will be DASD02.
  - The high level qualifier for the workspace data sets will be SYS1.RACF.
  - The LUNAME is MF2AP002.
  - You already have the following JCL to activate the RACF subsystem:

    ```
    //RACF PROC
    //RRSF EXEC PGM=IRRSSM00
    ```

  - You want to activate automatic direction for the node and you want the OUTPUT level of FAIL and the NOTIFY level of FAIL for the user IDs: SECADM and SYSPRG on MVS02 and SECADM on MVS01.

– APPC and VTAM have already been installed and configured.
– The JESNODE that will be used for transmits will be THATJES.

The following pages contain sample completed worksheets for this scenario.

Completed RRSF Node Configuration Worksheet for Node MVS01

Local Node:

Name: _MVS01_____

Type of Node:

____ Single-system ____ Multisystem

System Name: _____ Main: ___ Yes ___ No

Local node status:

_X__ Operative ____ Dormant

APPC protocol information:

LUNAME: _MF1AP001_____ (from SYS1.PARMLIB(APPCPM*xx*))
TPNAME: _____ (default=IRRRACF)
MODENAME: _____ (default=IRRMODE)

Workspace information:

Prefix name for high level qualifier for data set names: _SYS1.RACF_____
Filesize for workspace: _____ (initial=500)
____ **DFP SMS**
Name of SMS storage class: _____
Name of SMS data class: _____
Name of SMS management class: _____
_X__ **DFP Non-SMS**
Volume serial number to contain the workspace data sets: _DASD01_____

RACF parameter library information:

Name of

RACF parameter library data set:
_RRSF.PARM_____
Name of the member that is invoked automatically when the RACF subsystem initializes:
IRROPT_01___ (default=IRROPT00)

Password synchronization:

_X__ Yes ____ No

Command direction:

_X__ Yes ____ No

Automatic direction:

_X__ Yes ____ No
Output Level: _OUTPUT/FAIL_____ (ALWAYS | WARN | FAIL | NOOUTPUT)
Notify Level: _NOTIFY/FAIL_____ (ALWAYS | WARN | FAIL | NONOTIFY)

Node and user ID	Receive Output?	Notification?
_MVS01.SECADM___	__Y__	__Y__
_MVS01.SYSPRG___	__Y__	__Y__
_MVS02.SECADM___	__Y__	__Y__
_____	_____	_____

JESNODE: (for transmits)

_THISJES_____ (to override the value obtained automatically)

Remote NODE/SYSNAME:

Status of the local node's connection with each remote node:

_MVS02_____	_X__ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main

Completed RRSF Node Configuration Worksheet for Node MVS02

Local Node:

Name: _MVS02_____

Type of Node:

____ Single-system ____ Multisystem

System Name: _____ Main: ___ Yes ___ No

Local node status:

_X__ Operative ____ Dormant

APPC protocol information:

LUNAME: _MF2AP002_____ (from SYS1.PARMLIB(APPCPM*xx*))
TPNAME: _____ (default=IRRRACF)
MODENAME: _____ (default=IRRMODE)

Workspace information:

Prefix name for high level qualifier for data set names: _SYS1.RACF_____
Filesize for workspace: _____ (initial=500)
____ **DFP SMS**
Name of SMS storage class: _____
Name of SMS data class: _____
Name of SMS management class: _____
_X__ **DFP Non-SMS**
Volume serial number to contain the workspace data sets: _DASD02_____

RACF parameter library information:

Name of

RACF parameter library data set:
_RRSF.PARM_____
Name of the member that is invoked automatically when the RACF subsystem initializes:
IRROPT_02___ (default=IRROPT00)

Password synchronization:

_X__ Yes ____ No

Command direction:

_X__ Yes ____ No

Automatic direction:

_X__ Yes ____ No
Output Level: _OUTPUT/FAIL_____ (ALWAYS | WARN | FAIL | NOOUTPUT)
Notify Level: _NOTIFY/FAIL_____ (ALWAYS | WARN | FAIL | NONOTIFY)

Node and user ID	Receive Output?	Notification?
_MVS02.SECADM___	__Y__	__Y__
_MVS02.SYSPRG___	__Y__	__Y__
_MVS01.SECADM___	__Y__	__Y__
_____	_____	_____

JESNODE: (for transmits)

_THATJES_____ (to override the value obtained automatically)

Remote NODE/SYSNAME:	**Status of the local node's connection with each remote node:**			
_MVS01_____	_X__ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main
_____	____ Operative	____ Dormant	_____ Defined	_____ Main

Summary

This section contains a brief summary of steps that can be followed to transform the worksheet data into RACF configuration options. Each step is described in greater detail in the detailed instruction section that follows this summary.

1. Complete a configuration worksheet for each node (MVS01 and MVS02).
2. Create a RACF parameter library member IRROPT01 for node MVS01 with the TARGET and SET configuration information.
3. Create a RACF parameter library member IRROPT02 for node MVS02 with the TARGET and SET configuration information.
4. Modify the existing JCL that activates the RACF subsystem address space for node MVS01.
5. Modify the existing JCL that activates the RACF subsystem address space for node MVS02.
6. Issue the RACF STOP command to shut down the RACF subsystem on node MVS01.
7. Issue the RACF STOP command to shut down the RACF subsystem on node MVS02.
8. Issue the MVS START command on both nodes MVS01 and MVS02 to initialize the RACF subsystem address space.
9. Activate automatic direction on node MVS01.
10. Activate automatic direction on node MVS02.
11. Display the summary information for the network you have created using the TARGET command. Verify the information displayed for accuracy.
12. Display the detailed information for remote node MVS02 using the TARGET command from node MVS01. Verify the information displayed for accuracy.
13. Display the detailed information for local node MVS01 using the TARGET command from node MVS01. Verify the information displayed for accuracy.
14. Display the detailed information for remote node MVS01 using the TARGET command from node MVS02. Verify the information displayed for accuracy.
15. Display the detailed information for local node MVS02 using the TARGET command from node MVS02. Verify the information displayed for accuracy.
16. Display the attributes of the RRSF node MVS01 using the SET LIST command.
17. Display the attributes of the RRSF node MVS02 using the SET LIST command.
18. This two-node network is now configured for remote communication. For automatic command direction and automatic password direction to begin working, the appropriate automatic command direction and automatic password direction profiles must be defined and the RRSFDATA class must be activated. For command direction and password synchronization to begin working, user ID associations must be defined (via RACLINK), the appropriate command direction and password synchronization profiles must be defined, and the RRSFDATA class must be activated. See *z/OS Security Server RACF Security Administrator's Guide* for details.

Detailed Instructions

1. Complete a configuration worksheet for each node. See "Completed RRSF Node Configuration Worksheet for Node MVS01" on page 353 and "Completed RRSF Node Configuration Worksheet for Node MVS02" on page 354 for the completed worksheets.

2. On MVS01, edit the parameter library member IRROPT01 (from the RACF parameter library information section of the completed MVS01 worksheet) to include the following TARGET commands to establish MVS01 as a local node and to establish the communication link between MVS01 and MVS02:

   ```
   TARGET NODE(MVS01) DESCRIPTION('MEMPHIS

   MVS 1') - PREFIX(SYS1.RACF) LOCAL - WORKSPACE(VOLUME(DASD01)) -
   PROTOCOL(APPC(LUNAME(MF1AP001))) OPERATIVE TARGET
   NODE(MVS02) DESCRIPTION('ORLANDO MVS PROD') -
   PREFIX(SYS1.RACF) - WORKSPACE(VOLUME(DASD01)) -
   PROTOCOL(APPC(LUNAME(MF2AP002))) OPERATIVE
   ```

   As a result of running these TARGET commands, the following VSAM workspace data sets will be created for MVS01 on volume DASD01:
   - SYS1.RACF.SYSMVS01.INMSG
   - SYS1.RACF.SYSMVS01.OUTMSG
   - SYS1.RACF.MF1AP001.MF2AP002.INMSG
   - SYS1.RACF.MF1AP001.MF2AP002.OUTMSG

3. On MVS02, edit the parameter library member IRROPT02 (from the RACF parameter library information section of the completed MVS02 worksheet) to include the following TARGET command to establish MVS02 as a local node and to establish the communication link between MVS02 and MVS01:

   ```
   TARGET NODE(MVS02) DESCRIPTION('ORLANDO

   MVS PROD') - PREFIX(SYS1.RACF) LOCAL -
   WORKSPACE(VOLUME(DASD02)) -
   PROTOCOL(APPC(LUNAME(MF2AP002))) OPERATIVE TARGET
   NODE(MVS01) DESCRIPTION('MEMPHIS MVS 1') - PREFIX(SYS1.RACF) -
   WORKSPACE(VOLUME(DASD02)) -
   PROTOCOL(APPC(LUNAME(MF1AP001))) OPERATIVE
   ```

   As a result of running these TARGET commands, the following VSAM workspace data sets will be created for MVS02 on volume DASD02:
   - SYS1.RACF.SYSMVS02.INMSG
   - SYS1.RACF.SYSMVS02.OUTMSG
   - SYS1.RACF.MF2AP002.MF1AP001.INMSG
   - SYS1.RACF.MF2AP002.MF1AP001.OUTMSG

4. On MVS01, modify your existing JCL to activate the RACF subsystem to process the RACF parameter library. Add the PARM='OPT=01' parameter (from the RACF parameter library information section of the completed MVS01 worksheet) to the EXEC statement, to identify the member, and add a RACFPARM DD DSN=RRSF.PARM statement (from the RACF parameter library information section of the same worksheet) to identify the library.

   Your JCL on MVS01 should look like this:

   ```
   //RAC1     PROC
   //RRSF     EXEC PGM=IRRSSM00,PARM='OPT=01'
   //RACFPARM DD DSN=RRSF.PARM
   ```

5. On MVS02, modify your existing JCL to activate the RACF subsystem to process the RACF parameter library. Add the PARM='OPT=02' parameter (from the RACF parameter library information section of the completed MVS02 worksheet) to the EXEC statement, to identify the member, and add a RACFPARM DD DSN=RRSF.PARM statement (from the RACF parameter library information section of the same worksheet) to identify the library.

   Your JCL on MVS02 should look like this:

```
//RAC2     PROC
//RRSF     EXEC PGM=IRRSSM00,PARM='OPT=02'
//RACFPARM DD DSN=RRSF.PARM
```

6. Issue the RACF STOP command to shutdown the RACF subsystem on node MVS01, using the locally defined RACF subsystem prefix.

   ```
   @STOP
   ```

7. Issue the RACF STOP command to shutdown the RACF subsystem on node MVS02, using the locally defined RACF subsystem prefix.

   ```
   @STOP
   ```

8. Issue the MVS START command on both nodes MVS01 and MVS02 specifying the name of the RACF procedure to be started. The JCL will be read by MVS and the module will get control and complete the RACF subsystem address space initialization.

   ```
   START
   ```

   RACF,SUB=MSTR

9. Enter the following SET command from MVS01 to activate automatic direction on this node:

   ```
   @SET AUTODIRECT (OUTPUT (FAIL (MVS01.SECADM MVS01.SYSPRG MVS02.SECADM))
                   NOTIFY (FAIL (MVS01.SECADM MVS01.SYSPRG MVS02.SECADM)))
   ```

10. Enter the following SET command from MVS02 to activate automatic direction on this node:

    ```
    @SET AUTODIRECT (OUTPUT (FAIL (MVS02.SECADM MVS02.SYSPRG MVS01.SECADM))
                    NOTIFY (FAIL (MVS02.SECADM MVS02.SYSPRG MVS01.SECADM)))
    ```

11. Enter the following TARGET command from MVS01 to list the summary information for local node MVS01:

    ```
    @TARGET LIST
    ```

    You will receive the following output:

    ```
    IRRM009I (@) LOCAL RRSF NODE MVS01 IS IN THE OPERATIVE ACTIVE STATE.
    IRRM009I (@) REMOTE RRSF NODE MVS02 IS IN THE OPERATIVE ACTIVE STATE.
    ```

    Enter the following TARGET command from MVS02 to list the summary information for local node MVS02:

    ```
    @TARGET LIST
    ```

    You will receive the following output:

    ```
    IRRM009I (@) LOCAL RRSF NODE MVS02 IS IN THE OPERATIVE ACTIVE STATE.
    IRRM009I (@) REMOTE RRSF NODE MVS01 IS IN THE OPERATIVE ACTIVE STATE.
    ```

    If no error messages have been received during the set-up of these RRSF nodes, the connection state of these nodes should be operative active.

12. Enter the following TARGET command from MVS01 to list the detailed information for remote node MVS02:

    ```
    @TARGET NODE(MVS02) LIST
    ```

    You will receive the following output:

    ```
    IRRM010I (@) RAC1 SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE MVS02:
       STATE       - OPERATIVE ACTIVE
       DESCRIPTION - ORLANDO
    ```

    MVS PROD PROTOCOL - APPC LU NAME - MF2AP002 TP PROFILE NAME - IRRRACF MODENAME - <NOT SPECIFIED> TIME OF LAST TRANSMISSION TO - <NONE> TIME OF LAST TRANSMISSION FROM -

<NONE> WORKSPACE FILE SPECIFICATION PREFIX - ″SYS1.RACF″ FILESIZE - 500 VOLUME - DASD01 FILE USAGE ″SYS1.RACF.MF1AP001.MF2AP002.INMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S) ″SYS1.RACF.MF1AP001.MF2AP002.OUTMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S)

Verify that the detailed information provided is correct. If it is not, reissue the appropriate command or see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119 for more details.

13. Enter the following TARGET command from MVS01 to list the detailed information for local node MVS01:

```
@TARGET NODE(MVS01) LIST
```

You will receive the following output:

```
IRRM010I (@) RAC1 SUBSYSTEM PROPERTIES OF LOCAL RRSF NODE MVS01:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - MEMPHIS
```

MVS 1 PROTOCOL - APPC LU NAME - MF1AP001 TP PROFILE NAME - IRRRACF MODENAME - <NOT SPECIFIED> TIME OF LAST TRANSMISSION TO - <NONE> TIME OF LAST TRANSMISSION FROM - <NONE> WORKSPACE FILE SPECIFICATION PREFIX - ″SYS1.RACF″ FILESIZE - 500 VOLUME - DASD01 FILE USAGE ″SYS1.RACF.SYSMVS01.INMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S) ″SYS1.RACF.SYSMVS01.OUTMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S)

Verify that the detailed information provided is correct. If it is not, reissue the appropriate command or see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119 for more details.

14. Enter the following TARGET command from MVS02 to list the detailed information for remote node MVS01:

```
@TARGET NODE(MVS01) LIST
```

You will receive the following output:

```
IRRM010I (@) RAC2 SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE MVS01:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - MEMPHIS
```

MVS 1 PROTOCOL - APPC LU NAME - MF1AP001 TP PROFILE NAME - IRRRACF MODENAME - <NOT SPECIFIED> TIME OF LAST TRANSMISSION TO - <NONE> TIME OF LAST TRANSMISSION FROM - <NONE> WORKSPACE FILE SPECIFICATION PREFIX - ″SYS1.RACF″ FILESIZE - 500 VOLUME - DASD02 FILE USAGE ″SYS1.RACF.MF2AP002.MF1AP001.INMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S) ″SYS1.RACF.MF2AP002.MF1AP001.OUTMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S)

Verify that the detailed information provided is correct. If it is not, reissue the appropriate command or see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119 for more details.

15. Enter the following TARGET command from MVS02 to list the detailed information for local node MVS02:

```
@TARGET NODE(MVS02) LIST
```

You will receive the following output:

```
IRRM010I (@) RAC2 SUBSYSTEM PROPERTIES OF LOCAL RRSF NODE MVS02:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - ORLANDO
```

MVS PROD PROTOCOL - APPC LU NAME - MF2AP002 TP PROFILE NAME - IRRRACF MODENAME - <NOT SPECIFIED> TIME OF LAST TRANSMISSION TO - <NONE> TIME OF LAST TRANSMISSION FROM - <NONE> WORKSPACE FILE SPECIFICATION PREFIX - ″SYS1.RACF″ FILESIZE - 500 VOLUME - DASD02 FILE USAGE ″SYS1.RACF.SYSMVS02.INMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S) ″SYS1.RACF.SYSMVS02.OUTMSG″ - CONTAINS 0 RECORD(S) - OCCUPIES 1 EXTENT(S)

Verify that the detailed information provided is correct. If it is not, reissue the appropriate command or see Chapter 5, “RACF remote sharing facility (RRSF),” on page 119 for more details.

16. Enter the following SET command from MVS01 to list the attributes of the MVS01 node:

    ```
    @SET LIST
    ```

    You will receive the following output:

    ```
    IRRH005I (@) RAC1 SUBSYSTEM INFORMATION:
       TRACE OPTIONS                   - NOIMAGE
                                       - NOAPPC
       SUBSYSTEM USERID                - RRSFUSER
       JESNODE (FOR TRANSMITS)         - THISJES
       AUTOMATIC DIRECTION IS ALLOWED
             OUTPUT IS IN EFFECT FOR:
                     MVS01.SECADM      - FAIL
                     MVS01.SYSPRG      - FAIL
                     MVS02.SECADM      - FAIL
             NOTIFY IS IN EFFECT FOR:
                     MVS01.SECADM      - FAIL
                     MVS01.SYSPRG      - FAIL
                     MVS02.SECADM      - FAIL
    ```

    RACF STATUS INFORMATION: TEMPLATE VERSION - HRF7708 00000020.00000030 DYNAMIC PARSE VERSION - HRF7708

17. Enter the following SET command from MVS02 to list the attributes of the MVS02 node:

    ```
    @SET LIST
    ```

    You will receive the following output:

    ```
    IRRH005I (@) RAC2 SUBSYSTEM INFORMATION:
       TRACE OPTIONS                   - NOIMAGE
                                       - NOAPPC
       SUBSYSTEM USERID                - RRSFUSER
       JESNODE (FOR TRANSMITS)         - THATJES
       AUTOMATIC DIRECTION IS ALLOWED
             OUTPUT IS IN EFFECT FOR:
                     MVS02.SECADM      - FAIL
                     MVS02.SYSPRG      - FAIL
                     MVS01.SECADM      - FAIL
             NOTIFY IS IN EFFECT FOR:
                     MVS02.SECADM      - FAIL
                     MVS02.SYSPRG      - FAIL
                     MVS01.SECADM      - FAIL
    ```

RACF STATUS INFORMATION: TEMPLATE VERSION - HRF7708
00000020.00000030 DYNAMIC PARSE VERSION - HRF7708

18. This two-node network is now configured for remote communication. For automatic command direction and automatic password direction to begin working, the appropriate automatic command direction and automatic password direction profiles must be defined and the RRSFDATA class must be activated.

    For command direction and password synchronization to begin working, user ID associations must be defined (via RACLINK), the appropriate command direction and password synchronization profiles must be defined, and the RRSFDATA class must be activated. See *z/OS Security Server RACF Security Administrator's Guide* for details.

Now It's Your Turn to Fill Out the Worksheet

After reading the previous initialization scenario and following the detailed instructions, you should now be ready to fill out the node configuration worksheets for your network. The worksheet has been designed for you to make copies of it. It is suggested that you mark your installation-specific information regarding the RRSF network you wish to configure on these copies. Once completed, you should have all the data you need to define your RACF parameter libraries and configure your RRSF network.

Appendix C. Non-recommended Options

This appendix describes some RACF options that have been replaced with better product functions. These options are documented here for your information, but we do not recommend that you use them.

Selecting Options with ICHSECOP

The ICHSECOP module enables you to select the number of resident data blocks (when you don't have a data set name table, ICHRDSNT.) It enables you to bypass RACF-initialization processing (and RACF is inactive) and lets you disallow duplicate names for discrete data set profiles. These options are generally not needed and are not recommended. See Chapter 3, "RACF customization," on page 37 for options that are recommended.

This section describes the following options that you can specify in the ICHSECOP module:

- Bypassing RACF-initialization processing during IPL.
- Selecting the number of resident data blocks (only if there is no data set name table).

  Note: The preferred method of controlling the number of resident data blocks is to use the data set name table. If you have specified the number of resident data blocks in both the ICHSECOP module and the data set name table, RACF uses the figure in the data set name table.
- Disallowing duplicate names for data set profiles.

RACF contains a module (ICHSECOP) that you must replace in order to use these options. When you receive the module from IBM, it is set so that RACF-initialization processing is performed during IPL (and RACF is activated), ten data blocks are made resident, and duplicate data set profile names are allowed.

The module is used only during IPL. When you change the module, the changes are not effective until after the next IPL.

Module ICHSECOP contains 5 bytes of data, formatted as follows:

Byte 0	Bit 0	Bypass RACF-initialization processing (when set on)
	Bit 1	Disallow duplicate names for data set profiles (when set on)
	Bits 2-7	Reserved
Bytes 1-4		The number of data blocks to be made resident

Bypassing RACF-Initialization Processing

If you want to make RACF inactive, you can bypass RACF initialization processing during IPL by setting bit 0 of byte 0 on in module ICHSECOP. You can use this option as part of the procedure for bypassing RACF functions any time after the installation of RACF is complete.

This option (setting bit 0 on) makes RACF inactive until you turn bit 0 off and re-IPL. If this option is in effect (and RACF is inactive), you cannot use the RVARY command to make RACF active.

© Copyright IBM Corp. 1994, 2004

When this option is used, RACF does not verify a user's identity during TSO logon, IMS/VS or CICS/VS sign-on, or job-initiation processing. If a JOB statement contains the USER, GROUP, and PASSWORD parameters, the system ignores them. TSO reverts to UADS user identification and verification. Also, RACF commands cannot be issued.

If a user accesses a RACF-protected resource, the RACROUTE REQUEST=AUTH is still issued. If you are using any RACF-protected resources on your system, do the following:

- Use the SETROPTS command to turn off resource protection before bypassing RACF-initialization processing.
- Instruct the operations staff about the RACF failsoft messages and intervention requests.

The RACROUTE REQUEST=DEFINE is not issued by any RACF-related code in the system components unless failsoft processing allows the data set access and that data set is extended to a new volume. If you have written any modules using the RACROUTE REQUEST=DEFINE macro instruction, the failsoft processing in RACROUTE REQUEST=DEFINE gains control and issues messages to the system operator. The RACROUTE REQUEST=DEFINE failsoft processing also handles a job that has the PROTECT parameter specified on a DD statement. Note that RACROUTE REQUEST=DEFINE failsoft processing issues a message and continues normal processing without issuing an abend.

Selecting the Number of Resident Data Blocks

It is highly recommended that your installation have a data set name table (ICHRDSNT). You can use ICHRDSNT to specify the number of resident data blocks for each data set in the primary RACF database. (See "The data set name table" on page 37.)

If your installation does not have a data set name table, you can specify the number of resident data blocks for a single data set of the RACF database in ICHSECOP, or use the default value of 10 resident data blocks. However, be aware that using ICHRDSNT provides more flexibility and better performance options than using ICHSECOP.

If you have a data set name table and also have specified resident data blocks using ICHSECOP, the data set name table takes precedence during RACF processing.

You can select the number of RACF database data blocks to be made resident. An installation can specify from 0 to 255 resident data blocks; the default value is 10 resident data blocks. The blocks reside in ECSA.

Resident data blocks reduce the I/O processing that is required to service the RACF database. Each data block uses 4128 (4KB + 32) bytes of storage.

Disallowing Duplicate Names for Data Set Profiles

If you do not want your users to define duplicate data set names, turn on bit 1 of byte 0 in module ICHSECOP. (Duplicate data set names mean two discrete profiles have identical names, but reside on different volumes.)

If you choose this option, the RACF manager fails the ADDSD command and the RACF define macro if you attempt to define for a discrete data set profile a name that already exists.

Note: For RACF classes other than DATASET, you can never have duplicate profile names defined to RACF within the same class.

Changing the ICHAUTAB Module

The RACF authorized-caller table contains the names of programs that your installation authorizes to issue RACROUTE REQUEST=LIST or RACROUTE REQUEST=VERIFY without the NEWPASS keyword. The programs must be reentrant and fetched from an APF-authorized library.

Incorrect use of ICHAUTAB will cause system integrity problems; therefore, we recommend that you not use ICHAUTAB but rather run these programs with APF-authorization.

Using the RACF Authorized-Caller Table

Installation management must ensure that the programs it includes in the RACF authorized-caller table are both reentrant and protected so that users cannot modify code in these programs without prior review of the code by the installation management. Installation management should use RACF to protect their authorized libraries to ensure that only authorized users link edit programs into these libraries.

In addition, to avoid system-integrity problems, installation management must ensure that only authorized individuals, started tasks, or batch jobs are allowed to execute the programs whose names are included in the authorized-caller table.

You can use program control to control access to the program named in the authorized-caller table. For example, if you are using NCCF or NetView Release 1, you need to specify module DSIOST in the authorized-caller table. In addition, you should define DSIOST as a program with a universal access of NONE in the RACF PROGRAM class. Next, you should permit the NCCF or NetView started procedure to access the DSIOST program, using the user ID from the started procedures table. For this to work correctly, do not place DSIOST in the link pack area (LPA). Also, you should ensure that all copies of DSIOST that reside in APF-authorized libraries are defined to RACF and controlled by program control. You would continue to run NCCF or NetView Release 1 unauthorized, that is, with the linkage editor parameter of AC equal to 0.

If you are running NetView Release 2 or later, you should remove DSIOST from the authorized-caller table, as NetView Release 2 runs APF-authorized, and the entry is no longer necessary.

If you have placed the names of any other programs into the authorized-caller table, you should protect them using the approach outlined above. Additionally, it is highly recommended that you start planning to run those programs APF-authorized at your earliest convenience.

Format of the Authorized-Caller Table

For each authorized caller (program), the RACF authorized-caller table contains a 12-byte entry in the following format:

Length	Description

8 Caller name, left-justified and padded with blanks. (The last entry in the table must contain a blank caller name.)

4 Authorization code.

X'40000000' indicates that the caller is authorized to issue RACROUTE REQUEST=LIST.

X'80000000' indicates that the caller is authorized to issue RACROUTE REQUEST=VERIFY without the NEWPASS keyword.

The RACF authorized-caller table resides in the link pack area (LPA) in ICHAUTAB, which is an installation-replaceable module. To add an entry to the RACF authorized-caller table, you can do one of the following:

- Use the SPZAP service aid to add the entry to the ICHAUTAB module that IBM supplies. (See *z/OS MVS Diagnosis: Tools and Service Aids* for information on SPZAP.)

  Note: ICHAUTAB can handle up to six table entries. If your installation requires more than six, you must reassemble the ICHAUTAB module.
- Reassemble the ICHAUTAB module with the new entry and link edit it again into the LPA. You can link ICHAUTAB with either RMODE=24 or RMODE=ANY.

The following example shows how an installation can use the RACF authorized-caller table.

Example: An installation wants its VTAM application, Network Communications Control Facility (NCCF) (program number 5735-XX6) to be authorized to issue RACROUTE REQUEST=VERIFY without the NEWPASS keyword so that NCCF can perform additional security checks on the user's password. To do this, the installation codes the following in the RACF authorized-caller table:

Coded Description

CL8'DSIOST'
NCCF program name

XL4'80000000'
NCCF is authorized to issue calls to RACROUTE REQUEST=VERIFY without the NEWPASS keyword.

Note: This entry is not necessary for installations using NetView Release 2 and should be removed.

Appendix D. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS Information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

One exception is command syntax that is published in railroad track format; screen-readable copies of z/OS books with that syntax information are separately available in HTML zipfile form upon request to mhvrcfs@us.ibm.com.

© Copyright IBM Corp. 1994, 2004

Appendix E. Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

© Copyright IBM Corp. 1994, 2004

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

COPYRIGHT LICENSE:

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

This product contains code licensed from RSA Data Security Incorporated.

Programming Interface Information

This document is intended to help the RACF system programmer optimize and customize the RACF program product. It contains information about performance, installation exits, storage estimates, and operating considerations.

This publication documents intended programming interfaces that allow installations to write programs to obtain RACF services.

Trademarks

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

BookMaster
CICS
DB2
DFSMS/MVS
DFSMSdfp
DFSMSdss
DFSMShsm
DFSMSrmm
DFSORT
IBM
IBMLink
IMS
MVS
MVS/ESA
OS/390
Parallel Sysplex
PR/SM
RACF
Redbooks
Resource Link
SecureWay
VTAM
z/OS

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Lotus and Lotus Notes are trademarks of Lotus Development Corporation in the United States, other countries, or both.

TME, TME 10, and NetView are trademarks of International Business Machines Corporation or Tivoli Systems, Inc. in the United States, other countries, or both.

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

RSA is a trademark of RSA Data Security Incorporated.

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

Index

Special characters

??? state 134
* (asterisk)
  default for started procedures 98
  in the data set name table 38
  in the started procedures module 99
&RACUID.**/ALTER entry in the global access table 32
=
  in the started procedures module 99

Numerics

382 abend code 281, 282
383 abend code 299, 301
385 completion code 285, 286

A

abend codes
  382 281, 282
  383 299, 301
ACBNAME 159
access authority
  checking on job restarts 116
  checking with RACROUTE REQUEST=AUTH 279
  how it can cause accidental destruction of data 282
  using started procedures 94
access method service commands
  IMPORT command 113
  IMPORTRA command 113
  LISTCAT command 113
  REPRO command 113
  RESETCAT command 113
  using with RACF-protected VSAM data sets 113
accessibility 365
accidental deletion of data
  preventing 282
ACCTNUM class
  description 345
ACEE
  when there are multiple users per address space 116
ACEE (accessor environment element)
  compression/expansion exit routines 255
  default built by RACROUTE REQUEST=VERIFY 298
  when ICHRIX01 is responsible for creating 300
ACEEIEP 255
ACEERASP bit 252
ACICSPCT class
  description 341
activating
  automatic direction 154
  RACF databases 307
  the RACF subsystem 70
address space
  for sysplex data sharing 89
  multiple users 116
  RACF subsystem
    stopping 78
  RACFDS 89
ADDVOL operand
  RALTER command
    moving multivolume tape data sets 115
administering security 2
administration
  classroom courses xvi
ADRDSSU (DFDSS) system utility
  accessing a RACF-protected DASD data set 107
ADSP 127
ADSP (automatic data set protection) attribute
  when using IEHMOVE 109
AIMS class
  description 343
ALCSAUTH class
  description 339
algorithm
  Data Encryption Standard (DES) 54
  IBM Commercial Data Masking Facility (CDMF) 145
  masking 54
alias index blocks
  sample formatted output from IRRUT200 227
ALIGN keyword
  IRRUT400 utility 240
ALLOCATE system operation
  failures during 326
allocation
  BAM/allocation comparison 228
allocation authorization checks
  using the RACROUTE REQUEST=DEFINE exit routine to accept 283
ALTER command, access method services
  renaming RACF-indicated data sets 108
ALTER system operation
  failures during 326
AMASPZAP service aid
  changing the ICHRSMFI default values 59
APPC considerations for an RRSF network 145
APPCLU class
  description 339
APPCPORT class
  description 339
APPCSERV class
  description 339
APPCSI class
  description 339
APPCTP class
  description 339
APPL class
  description 339
application identity mapping
  recovering from errors 321
application updates, automatic direction of 125

© Copyright IBM Corp. 1994, 2004

association between user IDs
  overview 120
asterisk (*)
  in the data set name table 38
ASXBSENV pointer to ACEE 253
AT keyword on RACF commands 121
attributes of an RRSF node, listing 154
AUDIT operand
  ADDSD command 20
  ALTDSD command 20
  RALTER command 20
  RDEFINE command
    effect on system performance 20
  SETROPTS command
    effect on system performance 20
authorization checking 2
  performing additional checks for users 298
  using global access checking 32
authorized caller table 102
automatic command direction
  activating and deactivating 154
  commands issued from the RACF parameter library 169
  overview 123
automatic direction
  activating and deactivating 154
  order considerations 140
  overview 123
automatic direction of application updates
  activating and deactivating 154
  overview 125
automatic password direction
  activating and deactivating 154
  overview 124
automatic step restart 116

B

backing up a RACF database 8
backout routines
  commands that have 316
backup RACF database
  defining in the data set name table 37
  description 8
backup RACF databases
  creating 18
  defining in the data set name table 18
  effect when inactive in a sysplex 90
  levels of backup 18
  performance impact 18
BAM blocks
  BAM/allocation comparisons 228
  encoded map 228
    codes used in 228
  sample printout of an encoded map 230
basic RACF concepts 3
BCICSPCT class
  description 342
BLKUPD
  to correct problems with the RACF database 325
BLKUPD command 245
block alignment
  when using the IRRUT400 utility 233
block update command 245
BLP (bypass label processing)
  for tape volumes 114
browser for workspace data sets 246
buffers
  how the RACF manager keeps track of 41
  specifying in the data set name table 41

C

cache structure
  defining 90
  name for 91
  RACF support for REBUILDPERCENT 93
  reconfiguring 93
  size of 91
CACHECLS class
  description 339
cached auxiliary storage subsystem
  using for the RACF database 16
callers of exit routines
  summary of 253
CBIND class
  description 339
CCICSCMD class
  description 342
CDMF algorithm 139, 145
CDT (class descriptor table)
  changing an installation-defined class 51
  defining new classes 50
  deleting an installation-defined class 52
  description 48
  generating the table 52
  ICHERCDE macro 49
  ICHRRCDE module 49
  sysplex communication 49
  when the RACF database is shared 49
CFRM policy 90
change count in the ICB
  during processing on a shared RACF database 41
checking user authorization 2
CICS
  configuring the CICS timeout value range 105
  establishing defaults for using RACF 105
  general resource classes 341
CIMS class
  description 343
class
  changing 51
  defining for RACF/DB2 external security module 194
  defining new classes 50
  deleting 52
class descriptor table (CDT)
  changing an installation-defined class 51
  defining new classes 50
  deleting an installation-defined class 52
  description 48
  generating the table 52

class descriptor table (CDT) *(continued)*
ICHERCDE macro 49
ICHRRCDE module 49
supplied classes for z/OS and OS/390 systems 339
supplied classes for z/VM and VM systems 346
sysplex communication 49
when the RACF database is shared 49
class name
list of supplied general resource classes 339, 346
classification model I 191
classification model II 193
classroom courses, RACF xvi
CLPA parameter
for replacing the started procedures module 98
command direction
order considerations 140
overview 121
path through network 137
command direction, automatic
commands issued from the RACF parameter library 169
overview 123
command prefix
default 72
specifying 71
command prefix facility (CPF)
failed registration attempt 76
registering a command prefix with 71
commands
AT keyword 121
directing 121
exit routines for 261, 266
issued by exits 150
MVS START command 76
ONLYAT keyword 124
operator commands, RACF 79
RACLINK 120
RESTART 76
running in the RACF subsystem 79
sample exit to fail 271
SET 152
STOP 78
TARGET 155, 174
that do not modify RACF profiles 316
that have recovery routines 316
that perform multiple operations 318
that perform single operations 317
that propagate for RACF sysplex communication 322
to restart the RACF subsystem 76
with sensitive information 106
Commercial Data Masking Facility (CDMF) algorithm 145
completion code
385 285, 286
concepts, basic RACF 3
configuration worksheet, RRSF 349
connections between RRSF nodes
description 132
dormant 132
operative 132
connections between RRSF nodes *(continued)*
recycling 329
states 132
CONSOLE class
description 339
control points 94
control statements
for IRRUT100 utility 215
for IRRUT200 utility 222, 223
for IRRUT400 utility 236
control unit
for the RACF database 16
selecting for the RACF database 16
controlling access to resources 2
coordinator in a data sharing group
description 322
COPYAUTH parameter
when using IEHMOVE 110
copying a database
example using IRRUT400 242, 244
to a database with the exact same size 218
to a different device type 232
to a larger database
example using IRRUT400 243
to a larger or smaller database 232
to the same device type 218
using a two-stage process 244
using IRRUT200 218
using IRRUT400 232, 233
coupling facility
defining structures in CFRM policy 90
description 88
failures 311
RACF support for REBUILDPERCENT 93
rebuild support for RACF structures 314
reconfiguring RACF structures 93
sysplex recovery scenarios requiring non sysplex-communication/datasharing mode 315
sysplex recovery scenarios requiring XCF-local mode 314
using with a non-shared database 89
using with a single-system sysplex 89
courses about RACF xvi
CPSMOBJ class
description 342
CPSMXMP class
description 342
creating backup databases
using the IRRUT200 utility 18
using the IRRUT400 utility 18
cross-reference report
produced by IRRUT100 utility 212
CSA
storage requirement 337
CSFKEYS class
description 339
CSFSERV class
description 339
customization 37
assigning the data set in the database for a profile 45

customization *(continued)*
changing the RACF report writer options 57
CICS timeout value range 105
data set name table 37
database range table 45
defining resource classes 48
DES (Data Encryption Standard) algorithm for password authentication 54
duplicating updates on backup database 38
enabling sysplex communication 37, 39
enabling sysplex data sharing 37, 39
maintaining statistics on backup database 38
masking algorithm for password authentication 54
number of resident data blocks 38, 41
password authentication algorithm 54
RACF database options 37
RRSF environment 174
subsystem command prefix 71
CVTSNAME field 158

D

D-B state 133
D-E state 133
D-L state 133
D-R state 133
DASD data set 112
discrepancies between the profile and the indicator 318, 325
failures during system operations when RACF-protected 325
moving a data set with a discrete profile to a RACF-inactive system 111
moving a RACF-indicated data set to a non-RACF system 111
moving a RACF-indicated data set to a RACF-active system 111
not allowing duplicate profile names 362
operating considerations 106
renaming RACF-protected data sets 108
scratching 114
using access method service commands 113
using IEHMOVE with the ADSP attribute 109
using IEHMOVE with the COPYAUTH parameter 110
using the IMPORT command 113
using the IMPORTRA command 113
using the LISTCAT command 113
using the REPRO command 113
using the RESETCAT command 113
using utilities when RACF-protected 107
using utilities with the group-OPERATIONS attribute 108
using utilities with the OPERATIONS attribute 108
DASD device
selecting for the RACF database 16
DASD volume
moving between systems 114
operating considerations 113
scratching DASD data sets 114
DASDVOL authorization
operating considerations 113
DASDVOL class
description 339
data blocks
how resident blocks affect system performance 19
location of storage 41
size of 41
specifying in ICHSECOP 362
specifying resident blocks in the data set name table 38, 41
data masking in an RRSF network 145
data set name table
description 37
example of using 42, 43, 44
format of the flag field 38
selecting the number of resident data blocks 19
specifying the number of resident data blocks 41
specifying the RACF data set names 18
specifying the sysplex communication options 39, 89
data sharing mode
description 88
relationship to RRSF modes 88
database
activating 307
alternate database 8
authenticating the passwords and OIDCARD data 54
backing up 8
backup database 8
commands that modify only one profile 317
commands that perform multiple operations 318
considerations 1
copying 11
example of using IRRUT400 242, 243, 244
example using IRRUT200 223
to a database with a different size 233
to a database with same size 218
to a different device type 233
to same device type 218
using a two-stage process 244
using IRRUT200 utility 218
using IRRUT400 utility 233
creating 10
customizing 37
range table 45
DASD device 11
deactivating 307
determining percentage of space used 230
DFSMSdss DEFRAG and 13
discrepancies between profiles 315, 318
effect when backup inactive in a sysplex 90
encrypting the passwords and OIDCARD data 54
extending with IRRUT400 utility 233
failures during RACF manager processing 324
formatting with IRRMIN00 209
fragmentation 13
identifying inconsistencies using IRRUT200 218
index
scanning blocks 224

database *(continued)*
index *(continued)*
statistics from IRRUT200 225
initializing 208
insufficient space 309
insufficient space in 14
levels of backup 18
locating occurrences of a user ID or group name 212
location of 11
locking 234, 238
example of using IRRUT400 243
master primary data set 38
maximum number of data sets in 8
merging
example of using IRRUT400 243
modifying records in 245
monitoring usable space in 13
moving 11
multiple data sets 7
overview 4
prompting the operator for the data set name 38
quiescing 306
records initialized for a new database 210
recovery procedures 304
removing references to deleted IDs 245
reorganizing
using IRRUT400 utility 235
repairing
using IRRUT400 utility 235
residual authorities, removing 245
restoring 307
selection of control unit and device 16
sharing between systems 8
sharing data between remote systems 10
specifying options
data set name table 37
ICHSECOP module 361
splitting 7
example of using IRRUT400 243
storage requirement
factors affecting 333
formula 333
summary statistics from IRRUT200 230
switching 307
sysplex communication option 10
sysplex data sharing option 10
template level in use 64
templates
overview 5
updating 208
unloading to sequential file 245
unlocking 239
example of using IRRUT400 244
using resident index and data blocks 19
using the RVARY command 305
using utilities on
IRRIRA00 203
IRRMIN00 208
IRRUT100 212
IRRUT200 218
database *(continued)*
using utilities on *(continued)*
IRRUT400 232
summary 202
when to issue commands that update 21, 22
database range table
correspondence to the data set name table 46
description 45
example of using 47
location 45
when using IRRUT400 utility 236
database unload utility (IRRDBU00) 245
DATASET class
using a global access table for 32
DB2
general resource classes 342
protecting DB2 data 106
RACF/DB2 external security module 187
using RACF for authorization checking 187
DBSYNC EXEC 145
DCE
general resource classes 343
DCICSDCT class
description 342
DD statements (JCL)
for the IRRIRA00 utility 207
for the IRRMIN00 utility 211
ddname
for IRRUT400 input data sets 237
for IRRUT400 output data sets 237
deactivating
automatic direction 154
RACF databases 307
DEF state 133
deferred step restart 116
DEFINE system operation
failures during 326
defined state 133
DEFRAG
using with a RACF database 13
DELDSD command
when moving a data set with a discrete profile 111
when moving a RACF-indicated data set to a non-RACF system 111
DELETE keyword on TARGET command 165
DELETE system operation
failures during 326
DES (Data Encryption Standard) algorithm
replacing by using the ICHDEX01 exit routine 276
using 54
DESCRIPTION keyword on TARGET command 159
device
for the RACF database 16
UCB above 16MB 114
DEVICES class
description 339
DFP (Data Facility Product)
information in the RACF database 105
DFSMS (Data Facility Storage Management Subsystem)
information in the RACF database 105
DFSMS enhanced data integrity (EDI) 13

DFSMSdfp
general resource classes 344
DFSMSdss DEFRAG
using with a RACF database 13
diagnostic capability
IRRIRA00 206
IRRMIN00 211
IRRUT100 214
IRRUT200 219
Diagnostic Capability
IRRUT400 236
DIGTCERT class
description 339
DIGTCRIT class
description 339
DIGTNMAP class
description 339
DIGTRING class
description 340
DIMS class
description 343
DIRACC class
description 345
DIRAUTH class
description 340
directed command
automatic 123
order considerations 140
overview 121
path through network 137
directed password, automatic
overview 124
direction of application updates, automatic 125
DIRECTRY class
description 346
DIRSRCH class
description 345
disability 365
disabling RACF 63
discrete profile
changes when the data set is renamed 109
moving a data set with a discrete profile to a RACF-inactive system 111
rules for renaming a data set 108
updating with the correct volume serial number 113
using IEHMOVE with the COPYAUTH parameter 110
DLFCLASS class
description 340
documents, licensed xvii
dormant by local request state 133
dormant by mutual request state 133
dormant by remote request state 133
dormant connection 132
dormant in error state 133
DORMANT keyword on TARGET command 165
DSIOST module
with NCCF 365
with NetView 365
DSMON (data security monitor)
RACF exits report 250
DSNADM class
description 342
DSNR class
description 342
dumps
of the RACF database 8
DUPDATASETS keyword
IRRUT400 utility 240
duplicate data set names
disallowing in ICHSECOP 362
how IRRUT400 handles 240
dynamic allocation parameters
in the ICHRSMFI module 57
dynamic exits facility 266
dynamic parse
automating initialization of 65
determining the level in use 64
initializing 64
RRSF considerations 143
dynamic started procedures table 97

E

ECICSDCT class
description 342
ECSA
storage requirement 338
EDI (enhanced data integrity), DFSMS 13
EJBROLE class
description 343
ELSQA
storage requirement 337
enabling RACF 63
encoded map
of a BAM block 228
sample printout by IRRUT200 230
encryption 54
end-of-volume processing
failures during 327
enhanced data integrity (EDI), DFSMS 13
ENQ names used by RACF 81
Enterprise Java Beans
general resource classes 343
ENVR object
and ICHRFX03 exit 289
and ICHRFX04 exit 294
description 255
EOV processing
failures during 327
EPLPA
storage requirement 337
erase-on-scratch
how it affects system performance 23
ESQA
storage requirement 337
EXEC statement
for IRRIRA00 utility 207
for IRRMIN00 utility 211
PARM parameters when executing IRRUT400 238
exit routine 249
ACEE compression/expansion 255

exit routine *(continued)*
availability of started procedures name to 94
commands 266
examining during RACF failures 304
extended addressing 250
for RACF commands 261
how they affect system performance 24
ICHCCX00 261, 265
ICHCNX00 261, 262
ICHDEX01 275
ICHDEX11 275
ICHPWX01 271
ICHRCX01 280
ICHRCX02 281
ICHRDX01 284
ICHRDX02 285
ICHRFX01 286
ICHRFX02 291
ICHRFX03 288
ICHRFX04 293
ICHRIX01 299
ICHRIX02 300
ICHRLX01 297
ICHRLX02 297
ICHRSMFE 301
ICHRTX00 302
ICHRTX01 302
IRRACX01 255
IRRACX02 255
IRREVX01 266
IRRSXT00 302
naming convention table 251
new password 271
overview 249
password authentication 275
possible uses of
allowing access when RACF Is inactive 282
controlling access of a shared user ID 283, 295
modifying data set naming conventions 252
password quality control 274
protecting the user's resources from the user 282
RACF exits report from DSMON 250
RACROUTE REQUEST=AUTH 279
postprocessing 281
preprocessing 280
RACROUTE REQUEST=DEFINE 283
postprocessing 285
preprocessing 284
RACROUTE REQUEST=FASTAUTH 286
postprocessing 290, 291, 293
preprocessing 286, 288
RACROUTE REQUEST=LIST 296, 297
pre- and postprocessing 297
RACROUTE REQUEST=VERIFY(X) 298
postprocessing 300
preprocessing 299
report writer 301
requirements for 249
RRSF considerations 149
SAF callable services router 302
exit routine *(continued)*
SAF router 302
save area 249
selection 297
summary of callers 253
use of ASXBSENV pointer to ACEE 253
use of PUTLINE 252
use of TCBSENV pointer to ACEE 253
use of TPUT 252
use of WTO 252
exit-generated command 150
exit-generated update 150
exits report
from DSMON 250
extending a database
using IRRUT400 233
extents
when using the IRRUT400 utility 238
external security module, RACF/DB2 187

F

FACILITY class
description 340, 346
failsoft mode
description 89
failsoft processing 104
description 102
exits called 103
for RACROUTE REQUEST=DEFINE 362
general considerations 103
how it can affect system performance 22
impact on users 104
permanent 103
temporary 103
failures
coupling facility 311
during ALLOCATE or DEFINE operations 326
during EOV operation 327
during RACF command processing 315
during RACF manager processing 324
during RENAME or ALTER operations 326
during SCRATCH or DELETE operations 326
during system operations on data sets 325
failsoft processing 104
RACF parameter library 327
recovery procedures 304
shutting down the RACF subsystem 331
using user ID in SYS1.UADS to logon 305
VSAM 329
FCICSFCT class
description 342
FIELD class
description 340, 346
FILE class
description 346
FIMS class
description 343
flags
flag field in the data set name table 38

FLPA
storage requirement 337
FORCE command (MVS)
using RACF STOP command instead of 78
formatted output of the index blocks 225
fragmentation in the RACF database 13
FREESPACE keyword
IRRUT400 utility 240
using with IRRUT400 238
FSOBJ class
description 345
FSSEC class
description 345

G

GCICSTRN class
description 342
GCPSMOBJ class
description 342
GCSFKEYS class
description 340
GDASDVOL class
description 340
GDG (generation data group)
renaming individual data sets 108
GDSNBP class
description 342
GDSNCL class
description 342
GDSNDB class
description 342
GDSNJR class
description 342
GDSNPK class
description 342
GDSNPN class
description 342
GDSNSC class
description 342
GDSNSG class
description 342
GDSNSM class
description 342
GDSNSP class
description 342
GDSNSQ class
description 342
GDSNTB class
description 342
GDSNTS class
description 342
GDSNUF class
description 343
GDSNUT class
description 343
GEJBROLE class
description 343
general resource
definition 48
general resource class 342
general resource class *(continued)*
changing 51
changing the class descriptor table 52
defining new classes 50
product use of
CICS 341
DB2 342
DCE 343
DFSMSdfp 344
Enterprise Java Beans 343
IMS 343
Infoprint Server 343
Information/Management 344
LFS/ESA 344
License Manager 344
Lotus Notes for z/OS 344
MQSeries 344
NetView 344
Novell Directory Services for OS/390 344
Security Server Network Authentication Service 344
SMS 344
Tivoli 345
Tivoli Service Desk 344
TSO 345
z/OS UNIX 345
supplied 339, 346
generic entries
coding in the started procedures table 98, 99
generic profile
considerations when renaming data sets 109
during authorization checking 32
internal name for the range table 46
modified by RACF 46
performance considerations 32
when moving a RACF-indicated data set to a non-RACF system 112
generic profiles
effect on performance 33
GID mapping
improving performance 34
VLF class needed for 69
GIMS class
description 343
GINFOMAN class
description 344
global access checking
for bypassing normal RACROUTE REQUEST=AUTH processing 32
how it affects system performance 24
when moving a RACF-indicated data set to a non-RACF system 112
global access table
for the DATASET class 32
the entry &RACUID.**/ALTER 32
GLOBAL class
description 340, 346
global resource serialization 80
GLOBALAUDIT operand
RALTER command
effect on system performance 20

GMBR class
description 340, 346
GMQADMIN class
description 344
GMQNLIST class
description 344
GMQPROC class
description 344
GMQQUEUE class
description 344
group
information about provided by IRRUT100 utility 212
large, effect on performance 35
universal, effect on performance 35
group data set
preventing accidental destruction of data 282
group name
listing all occurrences on the RACF database 212
group profile 3
group tree in storage 79
activating 80
group-OPERATIONS attribute
determining the owner field when using IEHMOVE 110
when renaming a RACF-indicated data set 109
when using utilities 108
GRPACC (group access) attribute
when renaming a RACF-indicated data set 109
GSDSF class
description 340
GTERMINL class
description 340, 346
GTF traces 154

H

HCICSFCT class
description 342
HIMS class
description 343

I

I/O activity
RVARY command 306
I/O device
UCB above 16MB 114
IBM Commercial Data Masking Facility (CDMF) algorithm 145
ICB (inventory control block)
change count 41
RBAs of the templates defined 228
ICH408I message 114
ICH508I message 252
ICH522I message 100
ICH579E, message 6
ICH702A message 306
ICH703A message 306
ICHAUTAB module
adding entries 364
changing 102, 363
ICHAUTAB module *(continued)*
example of an entry 364
format of entries 363
ICHCCX00 exit routine
callers of 265
parameter list 265
return codes 266
uses of 261
when entered 262
ICHCNX00 exit routine
callers of 262
parameter fields available to 263
parameter fields that can be changed 263
parameter list 262
return codes 264
uses of 261
when entered 261
ICHDEX01 exit routine 276
ICHDEX11 exit routine 278
ICHERCDE macro 49
defining new classes in the class descriptor table 50
generating the class descriptor table 52
ICHNCV00 module 251
ICHPWX01 exit routine 271
conditions for gaining control 271
parameter list 272
return codes 273
ICHRCX01 exit routine
parameter list 280
preventing accidental destruction of data 283
return codes 280
when RACF is inactive 282
ICHRCX02 exit routine 281
return codes 282
what RACF does before it receives control 281
ICHRDSNT module
description 37
example of using 42, 43, 44
for specifying the RACF databases 18
selecting the number of resident data blocks 19
specifying the number of resident data blocks 41
specifying the sysplex communication options 39
ICHRDX01 exit routine 284
called during failsoft processing 103
requirements for 283
return codes 285
ICHRDX02 exit routine 285
requirements for 283
return codes 285
ICHRFR00 module
receiving control from the SAF router 94
ICHRFR01 module
description 52
ICHRFR0X module
description 94
ICHRFRTB macro
defining new entries in the router table 53
entries corresponding to the class descriptor table entries 50

ICHRFX01 exit routine
environment executed in 286
parameter list 286
requirements for 286
return codes 287
ICHRFX02 exit routine
environment executed in 291
parameter list 291
reason codes 292
requirements for 291
return codes 292
ICHRFX03 exit routine
environment executed in 288
parameter list 288
requirements for 286, 288
return codes 289
ICHRFX04 exit routine
environment executed in 293
parameter list 293
reason codes 295
requirements for 293
return codes 294
ICHRIN03 module
assigning a user ID to the RACF subsystem 74
coding 98
defining started procedures 98
description 98
examples of entries 101
format of the entries 98
* in the procedure name field 99
= for the user ID or group name 99
generic entries 98
ICHRIX01 exit routine 299
creating and initializing the ACEE 300
making password checks 274
return codes 299
ICHRIX02 exit routine 300
return codes 301
what RACF does before it receives control 300
ICHRLX01 exit routine
functions during RACROUTE REQUEST=LIST processing 296
requirements for 296
return codes 297
ICHRLX02 exit routine
functions during RACROUTE REQUEST=LIST processing 296
parameter list 297
requirements for 297
return codes 298
uses for 296
ICHRRCDE module 49
adding installation-defined classes 50
changing installation-defined classes 51
deleting installation-defined classes 52
ICHRRNG module
correspondence to the data set name table 46
description 45
example of using 47
location 45
relationship to IRRUT400 output data sets 237, 240
ICHRRNG module *(continued)*
when using IRRUT400 utility 236
ICHRSMFE exit routine
parameter list 302
return codes 302
uses of 301
when it is called 301
ICHRSMFI module
changing 57
format of 58, 59
ICHRTX00 exit 302
ICHRTX01 exit 302
ICHSECOP module
bypassing RACF initialization processing 361
description of options 361
disallowing duplicate names for DASD data set profiles 362
format 361
resident data blocks
selecting the number 362
ICHSFR00 module
description 93
ICKDSF (Device Support Facilities) system utility
accessing a RACF-protected DASD data set 107
identifying
RACF users 2
identity mapping profiles
recovering from errors 319
IEBCOMPR system utility
accessing a RACF-protected DASD data set 107
IEBCOPY system utility
accessing a RACF-protected DASD data set 107
IEBDG system utility
accessing a RACF-protected DASD data set 107
IEBEDIT system utility
accessing a RACF-protected DASD data set 107
IEBGENER system utility
accessing a RACF-protected DASD data set 107
use by IRRUT200 219
IEBISAM system utility
accessing a RACF-protected DASD data set 107
IEBPTPCH system utility
accessing a RACF-protected DASD data set 107
IEBUPDTE system utility
accessing a RACF-protected DASD data set 107
IEFSSNxx member of SYS1.PARMLIB 71
IEHATLAS system utility
accessing a RACF-protected DASD data set 107
IEHINITT system utility
restricting its use 115
IEHLIST system utility
accessing a RACF-protected DASD data set 107
IEHMOVE system utility
accessing a RACF-protected DASD data set 107
determining the owner 110
renaming RACF-indicated data sets 108
using with RACF-indicated DASD data sets 109
using with the ADSP attribute 109
using with the COPYAUTH parameter 110
IEHPROGM system utility
renaming RACF-indicated data sets 108

ILMADMIN class
description 344
IMPORT command
using on RACF-protected VSAM data sets 113
IMPORTRA command
using on RACF-protected VSAM data sets 113
IMS (Information Management System)
general resource classes 343
setting a pointer to the ACEE 116
in-storage profile
using RACROUTE REQUEST=LIST to build 296
inactive (RACF)
allowing access 282
by bypassing RACF initialization processing 361
index blocks
formatted output from IRRUT200 225
sample formatted output from IRRUT200 226
scanning with IRRUT200 utility 224
structure correction with IRRUT400 utility 233
unformatted output by IRRUT200 225
index compression
when using the IRRUT400 utility 233
index entries
problems due to failures during RACF manager processing 324
index structure
correcting when using IRRUT400 utility 233
INFOMAN class
description 344
Infoprint Server
general resource class 343
Information/Management
general resource classes 344
initial state 134
initialization processing
bypassing 361
initialization routine
loading the RACF exit routines 249
locating the naming convention table module 252
initialization worksheet, RRSF 349
INITSTATS option on SETROPTS 29
INITSTATS processing 38
INMSG data set for RRSF 134
installation exit routine 249
installation-defined class
adding 50
changing 51
deleting 52
installing RACF
formatting the RACF database 209
storage requirements 333
installing the REXX RACVAR function 117
insufficient space condition on the RACF database 309
internal names
how RACF constructs for the range table 46
inventory control block (ICB)
change count 41
IPCOBJ class
description 345
IPL
of a RACF data sharing group 40
use of the ICHSECOP module 361
IRR@XACS member of SYS1.SAMPLIB 188
IRR402I message 324
IRR403I message 324
IRR404I message 324
IRRACEE class 32
IRRACX01 and IRRACX02 exit routines 255
IRRADU00 utility 245
IRRBRW00 utility 246
IRRDBU00 utility 245
IRRDPI00 command
authorization 65
automating 65
syntax 65
IRRDPSDS data set
determining the level in use 64
RRSF considerations 143
IRRDPTAB started procedure 66
IRREVX01 exit point 266
IRREVX1A sample exit 271
IRREVX1B sample exit 271
IRRGTS class 80
IRRIRA00 utility
DD statements for 207
description 203
example 207
input 207
output 207
return codes 207
using 207
IRRMIN00 utility
comparison to IRRUT400 utility 238
DD statements for 211
description 208
input 211
output 212
return codes 212
using 211
IRRRID00 utility 245
IRRSEQ00 callable service
requirement for RACF subsystem 70
IRRSMAP VLF class 34, 69
IRRSXT00 exit 302
IRRTEMP2 5
IRRUT100 utility
associated exit routine 213
control 236
description 212
example 216
information provided by the report 212
input and output 215
job control statements 215, 236
sample output of the printed report 217
using 215
utility control statements 215
work data set 214
IRRUT200 utility 225
BAM/allocation comparison 228
control 222

IRRUT200 utility *(continued)*
description 218
example 223
for taking dumps of the RACF database 8
functions 218
identifying problems with the RACF database 325
input and output 222
job control statements 222
return codes 232
sample output
formatted alias index blocks 227
formatted index blocks 226
of the encoded map 230
unformatted index blocks 225
scanning the index blocks 224
using 221
using a copy to update the RACF database 22
utility control statements 222, 223
IRRUT400 utility
comparison to IRRMIN00 utility 238
considerations for keyword LOCKINPUT 239
copying a database 233
description 232
example of copying a database 242, 244
example of copying to larger database 243
example of locking a database 243
example of merging data sets 243
example of splitting a database 243
example of unlocking a database 244
examples of coding 242
executing 236, 240
input database specification 237
output data set processing 238
output data set selection 237
output database specification 237
parameter specification 238
processing of conflicts and inconsistencies 240
extending a database 233
how it works 233
parameters 238
reorganizing a database 235
repairing a database 235
return codes 241
when not to use 240
with LOCKINPUT keyword 18
ISMF
use of panel driver interface 116
ISPF
issuing commands from 106
ISPLOG data set
logging commands with sensitive information 106

J

JAVA class
description 343
JCICSJCT class
description 342
JCL (job control language)
examples of coding IRRUT400 utility 242
EXEC statement for IRRIRA00 207
JCL (job control language) *(continued)*
EXEC statement for IRRMIN00 211
for creating a database 11
for IRRUT100 utility 215
for IRRUT200 utility 222
for IRRUT400 utility 236
parameters ignored when bypassing RACF initialization 362
specifying bypass label processing 114
to activate the RACF subsystem 75
JES initialization
specifying bypass label processing 114
JESINPUT class
description 340
JESJOBS class
description 340
JESSPOOL class
description 340
JOB statement (JCL)
for started procedures 94
parameters ignored when bypassing RACF initialization 362
specifying the password when restarting jobs 116
jobs
restarting 116

K

KCICSJCT class
description 342
KERBLINK class
description 344
keyboard 365

L

LABEL parameter on DD statement
specifying bypass label processing 114
LAN File Services/ESA (LFS/ESA)
See LFS/ESA (LAN File Services/ESA)
large group, effect on performance 35
large profile, effect on performance 34
LFS/ESA (LAN File Services/ESA)
general resource class 344
LFSCLASS class
description 344
License Manager
general resource class 344
licensed documents xvii
LIST keyword on TARGET command 161
LISTCAT command
using on RACF-protected VSAM data sets 113
listing the attributes of an RRSF node 154
LOCAL keyword on TARGET command 159
local mode for an RRSF node
configuration example 176
description 131
relationship to sysplex communication modes 88
local node 129
local peer system 131
local system 131

location of resident data blocks 362
location of the RACF database 11
locking a database
  example using IRRUT400 243
LOCKINPUT keyword
  IRRUT400 utility 238
  using with IRRUT400 234
logging
  how it affects system performance 20
  using RACROUTE REQUEST=AUTH exit routine to modify 279
logging and reporting 2
LOGREC records for SETROPTS processing on a sysplex 324
LookAt message retrieval tool xvi
Lotus Notes for z/OS
  general resource class 344
LSQA
  storage requirement 337
LU name
  how to determine 159
LUNAME keyword on TARGET command 159

M

macros issued by exits 150
MAIN keyword on TARGET command 158
main system
  configuring a new one 167
  defining 158
  description 129
  selecting 152
maintenance
  restarting a function after applying 77
mapping UIDs and GIDs
  improving performance 34
  VLF classes needed for 69
masking algorithm 54, 275
master primary RACF data set 38
MCICSPPT class
  description 342
MDSNBP class
  description 343
MDSNCL class
  description 343
MDSNDB class
  description 343
MDSNJR class
  description 343
MDSNPK class
  description 343
MDSNPN class
  description 343
MDSNSC class
  description 343
MDSNSG class
  description 343
MDSNSM class
  description 343
MDSNSP class
  description 343
MDSNSQ class
  description 343
MDSNTB class
  description 343
MDSNTS class
  description 343
MDSNUF class
  description 343
MDSNUT class
  description 343
member systems 131
merging data sets
  example using IRRUT400 243
merging data sets in the database
  using IRRUT400 232
message ICH579E 6
message retrieval tool, LookAt xvi
messages
  ICH408I 114
  ICH508I 252
  ICH522I 100
  IRR402I 324
  IRR403I 324
  IRR404I 324
MGMTCLAS class
  description 344
mode
  data sharing 88
  failsoft 89
  local 131
  non–data sharing 88
  read-only 89
  remote 132
MODENAME keyword on TARGET command 160
moving a multivolume RACF-indicated DASD data set 112
moving a RACF-indicated DASD data set
  between systems 110
  to a non-RACF system with RACF indicator checking 111
  to a RACF-active system 111
  with a discrete profile to a RACF-inactive system 111
moving DASD volumes between systems 114
moving tape volumes
  between systems 115
  multivolume tape data sets 115
MQADMIN class
  description 344
MQCMDS class
  description 344
MQCONN class
  description 344
MQNLIST class
  description 344
MQPROC class
  description 344
MQQUEUE class
  description 344
MQSeries
  general resource classes 344

multi-subsystem class scope 193
multiple input data sets
  considerations when using IRRUT400 utility 233
multiple users per address space 116
multisystem node 129
multisystem RRSF node
  adding a system to 166
  configuring a new main system 167
  deleting a system from 166
  selecting the main system for 152
  system requirements 143
multivolume tape data set 115
MVS router
  See also ICHRTX00 exit
  See SAF router
MVS START command 76

N

naming convention table
  functions it should perform 251
  RRSF considerations 149
  use of 251
  when processing occurs 252
naming conventions
  changing the standard naming conventions 251
  for RRSF workspace data sets 134
  modifying with exits 252
  when defining DASD data set profiles 107
NCCF
  program control of 365
NCICSPPT class
  description 342
NDSLINK class
  description 344
NETCMDS class
  description 344
NETSPAN class
  description 344
NetView
  general resource classes 344
  program control of 365
network
  RRSF 128
  security of data in 145
Network Authentication Service
  general resource classes 344
network-qualified name
  and workspace data set name 135
  on TARGET command 159
new password exit routine 271
NEWPASS keyword on RACROUTE
 REQUEST=VERIFY
  RACF authorized-caller table 363
NOALIGN keyword
  IRRUT400 utility 240
NOCMDVIOL operand
  SETROPTS command
    effect on system performance 21
NODE keyword on TARGET command 157
NODES class
  description 340
nodes, RRSF
  breaking a connection with 165
  connection states 132
  connections between 132
  defining 156
  description 128
  dormant connection 132
  local 129
  local mode 131
  mismatches in definitions of 157
  multisystem 129
  naming 157
  operative connection 132
  recycling connections 329
  remote 129
  remote mode 132
  single-system 129
NODMBR class
  description 340
NODUPDATASETS keyword
  IRRUT400 utility 240
NOFREESPACE keyword
  IRRUT400 utility 240
NOLOCKINPUT keyword
  IRRUT400 utility 238
non-shared RACF database
  consideration when changing to shared 8
  processing of in-storage buffers 42
non-VSAM data set
  failures during ALLOCATE operation 326
  failures during EOV system operation 327
  failures during RENAME operation 326
  failures during SCRATCH operation 326
  formatting for use as a RACF database 209
  turning off the RACF indicator and deleting the profile 111
  turning off the RACF indicator and preserving the profile 112
  when extended on one system and moved to another 112
non–data sharing mode 312
  description 88
  relationship to RRSF modes 88
nonmain system 129
NOSAUDIT operand
  SETROPTS command
    effect on system performance 21
NOSET operand
  ADDSD command
    when moving a RACF-indicated data set 111
not defined state 134
NOTABLE keyword
  IRRUT400 utility 239
NOTELINK class
  description 344
Notices 367
Novell Directory Services for OS/390
  general resource class 344

NVASAPDT class
description 344

O

O-A state 133
O-E state 133
O-P-C state 133
O-P-V state 133
OIDCARD data
authenticating algorithms 54
checking the validity with RACROUTE REQUEST=VERIFY 298
OIMS class
description 343
ONLYAT keyword on RACF commands 124
OPEN macro instruction
RACF authorization checking 107, 115
operating considerations 62
commands issued from ISPF 106
DASD data sets 106
for DASD volumes 113
moving a data set with a discrete profile to a RACF-inactive system 111
moving a multivolume RACF-indicated data set between systems 112
moving a RACF-indicated DASD data set between systems 110
moving a RACF-indicated DASD data set to a non-RACF system 111
moving a RACF-indicated DASD data set to a RACF-active system 111
moving DASD volumes between systems 114
moving tape volumes between systems 115
multiple users per address space 116
protecting DB2 data 106
RACF panel driver interface 116
renaming RACF-protected data sets 108
restarting jobs 116
REXX RACVAR function 117
scratching DASD data sets 114
tape volumes 114
bypass label processing 114
protection for unlabeled tapes 115
TSO profiles in the RACF database 106
UCBs above 16MB 114
using access method service commands 113
IMPORT command 113
IMPORTRA command 113
LISTCAT command 113
REPRO command 113
RESETCAT command 113
using IEHMOVE with the ADSP attribute 109
using IEHMOVE with the COPYAUTH parameter 110
using NCCF 365
using NetView 365
using the RACF authorized-caller table 365
using utilities on RACF-protected DASD data sets 107
using utilities on RACF-protected tape volumes 115
operating considerations *(continued)*
using utilities with the group-OPERATIONS attribute 108
using utilities with the OPERATIONS attribute 108
operating system and RACF interaction 3
OPERATIONS attribute
determining the owner field when using IEHMOVE 110
when RACF is inactive 282
when renaming a RACF-indicated data set 109
when using utilities 108
operative active state 133
operative connection 132
operative in error state 133
OPERATIVE keyword on TARGET command 164, 165
operative pending connection state 133
operative pending verification state 133
operator commands, RACF 79
operator prompts
during failsoft processing 104
for an asterisk in the data set name table 38
OPERCMDS class
description 340
options 37
changing the ICHAUTAB module 102, 363
changing the ICHRSMFI module 57
changing the RACF report writer options 57
CICS timeout value range 105
data set name table 37
database range table 45
defining resource classes 48
DES (Data Encryption Standard) algorithm for password authentication 54
duplicating updates on backup database 38
enabling sysplex communication 39
enabling sysplex data sharing 39
ICHDEX01 exit 54
maintaining statistics on backup database 38
masking algorithm for password authentication 54
number of resident data blocks 38, 41
password authentication algorithm 54
RRSF environment 174
specifying RACF database options 37
data set name table 37
database range table 45
ICHSECOP module 361
subsystem command prefix 71
using the system authorization facility (SAF) 93
which data set to place each profile on 45
OUTMSG data set for RRSF 134
OWNER field
resolving conflicts with RACROUTE REQUEST=LIST selection exit routine 297
owner of the profile
when using IEHMOVE with the COPYAUTH parameter 110

P

panel driver interface
operating considerations 116

parameter library, RACF
attributes 169
automatically processing at initialization 170
blank lines in 170
commands that can be issued from 169
comments in 170
configuring RRSF without 169
continuing commands in 169
description 168
initializing dynamic parse from 65
member names 169
order of commands in 174
processing using SET INCLUDE 171
recovering from errors 327
running IRRDPI00 from 65
security for 169
sharing 173
parameter lists
ICHCCX00 exit routine 265
ICHCNX00 exit routine 262
ICHPWX01 exit routine 272
ICHRCX01 exit routine 280
ICHRFX01 exit routine 286
ICHRFX02 exit routine 291
ICHRFX03 exit routine 288
ICHRFX04 exit routine 293
ICHRLX02 exit routine 297
ICHRSMFE exit routine 302
modifying with the SAF router exit routine 302
PARM field of the EXEC statement
parameters when executing IRRUT400 238
PassTicket
validating 56
password
authenticating algorithms
DES (Data Encryption Standard) 54, 275
installation-provided 275
masking 54, 275
migrating to the RACF DES algorithm 54
two-step method 54
authentication exit routines 275
checking validity with RACROUTE REQUEST=VERIFY 298
encryption 54
for activating or deactivating RACF 305
for changing the RACF operating mode 305
for RVARY 305
for switching the RACF database 305
new password exit routine 271
PassTicket as an alternative for 56
processing 56
quality control 274
use by RACF 2
when restarting jobs 116
PASSWORD command
invoking the new password exit routine 271
making password checks 274
password direction, automatic
overview 124
password rules
RRSF considerations 144
PCICSPSB class
description 342
PERFGRP class
description 345
performance
effect of large groups on 35
effect of large profiles on 34
effect of universal groups on 35
effect of UNIXMAP class 34
factors affecting the system 16
generic profiles 33
how erase-on-scratch can affect 23
how exit routines affect 24
how global access checking affects 24
how logging affects 20
how RACF commands can affect 21
how RACROUTE REQUEST=AUTH processing affects 32
how RACROUTE REQUEST=FASTAUTH processing affects 33
how RACROUTE REQUEST=VERIFY processing affects 31
how SETROPTS GENLIST processing affects 25
how SETROPTS RACLIST processing affects 25
how statistics gathering affects 29
how UID and GID mapping affects 34
how utility programs affect 22
using resident index and data blocks 19
VLF considerations 34
z/OS UNIX System Services applications 34
permanent failsoft 103
persistent verification
requirement for RACF subsystem 70
PIMS class
description 343
PLPA
storage requirement 337
PMBR class
description 340
policy, CFRM 90
POSIT number
effect on SETROPTS STATISTICS 30
precaution when changing 51
precaution when deleting a class 52
postprocessing exit routine
ICHRCX02 281
ICHRDX02 285
ICHRIX02 300
RACROUTE REQUEST=AUTH 281
return codes 282
RACROUTE REQUEST=DEFINE 285
requirements for 283
return codes 285
RACROUTE REQUEST=FASTAUTH
environment executed in 291, 293
parameter list 291, 293
reason codes 292, 295
requirements for 291, 293
return codes 292, 294
RACROUTE REQUEST=LIST
requirements for 296

postprocessing exit routine *(continued)*
RACROUTE REQUEST=LIST *(continued)*
return codes 297
RACROUTE REQUEST=VERIFY
return codes 301
RACROUTE REQUEST=VERIFY(X) 300
prefix
specifying for operator commands 71
PREFIX keyword on TARGET command 160
preprocessing exit routine
how system performance is affected 24
ICHCCX00
parameter list 265
return codes 266
ICHCNX00
calling before IRRUT100 utility 213
parameter list 262
return codes 264
ICHRDX01 284
ICHRIX01 299
RACROUTE REQUEST=AUTH
parameter list 280
return codes 280
RACROUTE REQUEST=DEFINE 284
requirements for 283
return codes 285
RACROUTE REQUEST=FASTAUTH
environment executed in 286, 288
parameter list 286, 288
requirements for 286, 288
return codes 287, 289
RACROUTE REQUEST=LIST
requirements for 296
return codes 297
RACROUTE REQUEST=VERIFY
return codes 299
RACROUTE REQUEST=VERIFY(X) 299
what RACF does before exit receives control 249
primary RACF database
defining in the data set name table 37
Print Services Facility (PSF)
See PSF (Print Services Facility)
PRINTSRV class
description 344
privileged attribute
for started procedures 95
PROCACT class
description 345
PROCESS class
description 345
profile
commands that do not modify 316
discrepancies between 318
discrepancies with indicator for DASD data sets 325
large, effect on performance 34
not allowing duplicate DASD data set names 362
specifying in ICHSECOP 362
PROGRAM class
description 340
program control 365
PROPCNTL class
description 341
PROTECT line operator in ISMF
use of panel driver interface 116
PROTECT parameter (JCL DD statement)
when restarting jobs 116
PROTECT=YES 127
protected user ID 74, 94
PROTOCOL keyword on TARGET command 159
PSF (Print Services Facility)
general resource class 341
PSFMPL class
description 341, 346
PTKTDATA class
description 341, 346
PTKTVAL class
description 344, 346
publications
on CD-ROM xv, xvi
softcopy xv, xvi
PURGE keyword on TARGET command 165
PUTLINE 252

Q

QCICSPSB class
description 342
QIMS class
description 343
quiescing RACF database I/O activity 306

R

R_admin callable service
requirement for RACF subsystem 70
RACF
bypassing initialization processing 361
classroom courses xvi
disabling 63
enabling 63
operating considerations 62
performance considerations 16, 25, 31
publications
on CD-ROM xv, xvi
softcopy xv, xvi
recovery procedures 104
utilities
IRRIRA00 203
IRRMIN00 208
IRRUT100 212
IRRUT200 218
IRRUT400 232
summary 202
RACF administration
classroom courses xvi
RACF authorized-caller table 102, 365
RACF commands
AT keyword 121
directing 121
exit routines for 261
ICHCCX00 265

RACF commands *(continued)*
exit routines for *(continued)*
ICHCNX00 261, 262
failures during RACF command processing 315
how they can affect system performance 21
ONLYAT keyword 124
operator commands 79
RACLINK 120
RESTART 76
running in the RACF subsystem 79
SET 152
STOP 78
TARGET 155
that do not modify RACF profiles 316
that have recovery routines 316
that perform multiple operations 318
that perform single operations 317
that propagate for RACF sysplex communication 322
RACF database
See database
RACF exits report
from DSMON 250
RACF indicator
turning off 111
RACF manager
failures during processing 324
processing of in-storage buffers 41
return code of 28 46
return code of 60 46
RACF options 37
RACF parameter library
attributes 169
automatically processing at initialization 170
blank lines in 170
commands that can be issued from 169
comments in 170
configuring RRSF without 169
continuing commands in 169
description 168
initializing dynamic parse from 65
member names 169
order of commands in 174
processing using SET INCLUDE 171
recovering from errors 327
running IRRDPI00 from 65
security for 169
sharing 173
RACF PROC
sample 75
RACF remote sharing facility
See RRSF (RACF remote sharing facility)
RACF report writer 245
default values in the ICHRSMFI module 57
description 57
exit routine 301
format of the ICHRSMFI module 58, 59
list of functions 57
options in the ICHRSMFI module 57
where documented xv
RACF router
receiving control from the SAF router 94
RACF router table
description 94
RACF security topics
classroom courses xvi
RACF subsystem
activating 70
assigning a user ID to 74
description 69
multiple 70
parameter library problems 327
recovery procedures 327
recovery when a task stops 328
restarting 76
restarting a function in 76
running commands in 79
sample JCL to activate 75
shutting down 331
specifying the command prefix for 71
RACF subsystem address space
stopping 78
RACF-indicated DASD data sets
moving a multivolume data set between systems 112
moving between systems 110
moving to a non-RACF system with RACF indicator checking 111
moving to a RACF-active system 111
RACF-protected indicator
discrepancies with DASD data set profiles 318, 325
RACF/DB2 external security module
assembling and link-editing 195
class scope 190
classification models 190
customizing 189
defining classes for 194
description 187
functions 196
installing 188
multi-subsystem class scope 190
single subsystem class scope 190
XAPLFUNC function codes 196
RACFDS address space 89
RACFICE reporting tool 246
RACFRCVY (RACF recovery) started procedure 308
RACFRW utility 245
RACFVARS class
description 341, 346
RACGLIST class
description 341
RACLINK command 120
RACROUTE macro
invoking the SAF router 302
RACROUTE REQUEST=AUTH
effect on system performance 32, 33
exit routines 279, 280
uses for 279
preprocessing routine called during failsoft 103
privileged attribute for started procedures 95
trusted attribute for started procedures 95

RACROUTE REQUEST=DEFINE
exit routines 283, 284, 285
preprocessing routine called during failsoft 103
when RACF initialization is bypassed 362
RACROUTE REQUEST=FASTAUTH
exit routines 286, 288, 291, 293
use of resident profiles 33
RACROUTE REQUEST=LIST
description of processing 296
exit routines 297
RACF authorized-caller table 363
RACROUTE REQUEST=VERIFY
building the default ACEE 298
effect of on system performance 31
RACF authorized-caller table 363
RACROUTE REQUEST=VERIFY CREATE
for multiple users per address space 116
RACROUTE REQUEST=VERIFY(X)
exit routine 298
exit routines 298
RACVAR function for REXX execs
information it provides 117
installing 117
range table 45
correspondence to the data set name table 46
example of using 47
location 45
relationship to IRRUT400 output data sets 240
when using IRRUT400 utility 236
range table for ACEE compression and expansion 256
RBA (relative byte address)
of a BAM block 228
of the templates defined in the ICB 228
read-only mode 311
description 89
relationship to RRSF modes 88
running IRRUT400 in 238
REALM class
description 344
reason codes
from ICHRFX02 exit routine 292
from ICHRFX04 exit routine 295
rebuild of RACF cache structures 93
rebuild support for RACF structures 314
REBUILDPERCENT 93
recovery procedures
for coupling facility failures 311
for failures during ALLOCATE or DEFINE operations 326
for failures during EOV operation 327
for failures during RACF command processing 315
for failures during RACF manager processing 324
for failures during RENAME or ALTER operations 326
for failures during SCRATCH or DELETE operations 326
for failures during system operations on data sets 325
for hung connection 329
for RACF parameter library problems 327
for the RACF database 304
recovery procedures *(continued)*
RACFRCVY started procedure 308
sample procedures 309
shutting down the RACF subsystem 331
synchronization considerations 307
TSO considerations 305
using UADS user ID 305
VSAM failures on the workspace data sets 329
when a task stops 328
when the workspace data sets fill up 329
recovery routines
commands that have 316
recycling a connection 329
REFRESH GENERIC operands
SETROPTS command 28
REFRESH RACLIST operands
SETROPTS command 27
remote mode for an RRSF node
configuration example 178
description 132
relationship to sysplex communication modes 88
remote node 129
remote sharing
See RRSF (RACF remote sharing facility)
remove ID utility (IRRRID00) 245
RENAME command, TSO
renaming RACF-indicated data sets 108
RENAME on a MOVE statement
when using IEHMOVE 109, 110
RENAME system operation
failures during 326
renaming RACF-indicated data sets 108
individual data sets of a GDG 108
reorganizing a database
using IRRUT400 235
repairing a database
using IRRUT400 235
replaceable modules
ICHAUTAB 363
ICHRDSNT 37
ICHRIN03 98
ICHRRNG 45
ICHRSMFI 57
ICHSECOP 361
report
produced by IRRUT100 utility 212
sample output from IRRUT100 217
report writer
default values in the ICHRSMFI module 57
description 57
exit routine 301
format of the ICHRSMFI module 58, 59
list of functions 57
options in the ICHRSMFI module 57
where documented xv
REPRO command
using on RACF-protected VSAM data sets 113
RESERVE macro
issued by IRRUT200 utility 221
RESETCAT command
using on RACF-protected VSAM data sets 113

resident data blocks
how they affect system performance 19
specifying in ICHSECOP 362
specifying in the data set name table 38, 41
resident index blocks
how they affect system performance 19
resident profiles
use by RACROUTE REQUEST=FASTAUTH for authorization checking 33
using RACROUTE REQUEST=LIST to build 296
ways used by RACF 281
resource class
changing 51
class descriptor table 48
defining for RACF/DB2 external security module 194
defining new classes 50
deleting 52
resource groups
during RACROUTE REQUEST=LIST processing 296
resource managers
using the RACROUTE REQUEST=AUTH 279
RESTART command 76
using after applying maintenance 77
using to recover from failures 78
restarting
functions in the RACF subsystem 76
jobs 116
the RACF subsystem 76
return codes
28 from the RACF manager 46
60 from the RACF manager 46
from ICHCCX00 exit routine 266
from ICHCNX00 exit routine 264
from ICHPWX01 exit routine 273
from ICHRCX01 exit routine 280
from ICHRCX02 exit routine 282
from ICHRDX01 exit routine 285
from ICHRDX02 exit routine 285
from ICHRFX01 exit routine 287
from ICHRFX02 exit routine 292
from ICHRFX03 exit routine 289
from ICHRFX04 exit routine 294
from ICHRIX01 exit routine 299
from ICHRIX02 exit routine 301
from ICHRLX01 exit routine 297
from ICHRLX02 exit routine 298
from ICHRSMFE exit routine 302
from IRRIRA00 utility 207
from IRRMIN00 utility 212
from IRRUT200 utility 232
from IRRUT400 utility 241
REXX RACVAR function
information it provides 117
installing 117
RMTOPS class
description 344
RODMMGR class
description 344
ROLE class
description 345
router table
defining new entries 53
description 52
RRSF (RACF remote sharing facility) 120
APPC considerations 145
application updates, automatic direction of 125
associations between user IDs 120
AT keyword 121
automatic command direction 123
automatic direction 123
activating and deactivating 154
automatic direction of application updates 125
automatic password direction 124
breaking a connection with a node 165
command direction
overview 121
command direction, automatic 123
configuration examples 176
configuration worksheet 349
configuring an RRSF network 152, 174
configuring without the RACF parameter library 169
connections between nodes
description 132
recycling 329
states 132
considerations for new-password exit 271
controlling
incoming requests 165
outgoing requests 164
customizing 174
data masking 145
defined state 133
defining nodes 156
defining standard sequences of configuration commands 168
directed command
order considerations 140
overview 121
path through network 137
directed command, automatic 123
directed password, automatic 124
direction of application updates, automatic 125
direction, automatic 123
dormant by local request state 133
dormant by mutual request state 133
dormant by remote request state 133
dormant connection 132
dormant in error state 133
dynamic parse version considerations 143
examples of configuration 176
exit considerations 149
initial state 134
initialization scenario 351
initialization worksheet 349
INMSG data set 134
installation exit considerations 149
installation-provided code considerations 151
introduction 10
IRRBRW00 246

RRSF (RACF remote sharing facility) *(continued)*
IRRDPSDS 143
listing attributes of target nodes 161
listing the attributes of an RRSF node 154
local mode 131
local node 129
local peer system 131
local system 131
member systems 131
multisystem node 129
naming convention table considerations 149
network 128
node name restrictions 157
nodes
defining 156
description 128
local 129
mismatches in definitions of 157
multisystem 129
remote 129
single-system 129
not defined state 134
ONLYAT keyword 124
operative active state 133
operative connection 132
operative in error state 133
operative pending connection state 133
operative pending verification state 133
OUTMSG data set 134
overview of functions 121
parameter library 168
password direction, automatic 124
password rule considerations 144
password synchronization 122
prerequisites 143
purging workspace data sets 165
RACF parameter library 168
RACF subsystem address space considerations 151
RACLINK command 120
recovery from parameter library problems 327
recovery when a task stops 328
remote mode 132
remote node 129
RRSFDATA class 174, 176
RRSFLIST data set 121
security for 176
SET command 152
SETROPTS options considerations 144
shared database considerations 8
single-system node 129
states of connection 132
STOP command considerations 79
subsystem address space considerations 151
TARGET command 155
target nodes, defining 156
template version considerations 143
tracing APPC and IMAGE events 154
user ID associations 120
VSAM file browser 246
VTAM considerations 145
RRSF (RACF remote sharing facility) *(continued)*
worksheet 349
workspace data sets
defining 135
description 134
determining how full 136
increasing the size of 329
maintaining 136
preallocating 135
recovering when they fill up 329
size guidelines 136
RRSFDATA class
customizing the RRSF environment with 174
description 341
establishing security for the RRSF environment with 176
RRSFLIST data set 121, 155
RVARSMBR class
description 341, 346
RVARY command
ACTIVE operand 307
failures when propagating 322
I/O activity 306
INACTIVE operand 307
password for 305
SWITCH operand 307
using with shared RACF database 305
RVARYPW operand on SETROPTS command 305

S

SAF (system authorization facility)
callable services router exit 302
invoking the MVS router 93
invoking the SAF router 302
using 93
SAF router 302
description 93
exit routine 302
samplib
IRR@XACS member 188
RACF/DB2 external security module 188
save area for installation exits 249
scanning index blocks
formatted printout of from IRRUT200 225
unformatted printout of by IRRUT200 225
with IRRUT200 utility 224
SCDMBR class
description 341, 346
SCICSTST class
description 342
SCRATCH system operation
failures during 326
scratching DASD data sets 114
SDSF (System Display and Search Facility)
general resource class 341
SDSF class
description 341
SECDATA class
description 341, 346

SECLABEL class
description 341, 346
SECLMBR class
description 341
security
administering 2
security administrator
role of 2
security requirements
how RACF meets 1
Security Server (RACF)
disabling 63
enabling 63
security topics for RACF
classroom courses xvi
security-sensitive fields in a user's profile 68
serialized access to shared databases 80
serializing access to resources 80
SERVAUTH class
description 341
SERVER class
description 341
SET command 152
INCLUDE keyword 171
SETROPTS command
failures when propagating 323
REFRESH GENERIC operands 28
REFRESH RACLIST operand 27
specifying rules for passwords 274
SETROPTS GENLIST processing
effect on system performance 25
SETROPTS INITSTATS 38
SETROPTS options
RRSF considerations 144
SETROPTS RACLIST processing
effect on system performance 25
SFSCMD class
description 346
shared RACF database 8
caution for class descriptor tables 49
consideration when changing to non-shared 8
in an RRSF network 129
processing of in-storage buffers 41
specifying the resident data block option 41
sysplex communication 84, 93
sysplex communication option 10
sysplex data sharing option 10
using the RVARY command 305
shared RACF parameter library 173
shared user ID
controlling access 283, 295
shortcut keys 365
SIMS class
description 343
single subsystem class scope 191
single-system node 129
SMESSAGE class
description 341
SMF data
when restoring a RACF database 308
SMF data unload utility 245
SMF records
when ICHRSMFE exit routine is called 301
SMS
general resource classes 344
SOMDOBJS class
description 341
SORT/MERGE parameters
in the ICHRSMFI module 57
space used in RACF database
determining 230
SPECIAL authority
sample exit to limit 271
splitting a database
example using IRRUT400 243
using IRRUT400 232
SPZAP service aid
adding entries to ICHAUTAB 364
SQA
storage requirement 337
standard naming convention for data set profiles 107
START command, MVS 76
STARTED class
assigning a user ID to the RACF subsystem 75
description 341
using 97
started procedure
description 94
initializing dynamic parse from 66
privileged attribute 95
running IRRDPI00 from 66
trusted attribute 95
started procedures table
assigning a user ID to the RACF subsystem 74
coding 98
defining started procedures 98
description 98
examples of entries 101
generic entries 98
states of connection for RRSF nodes 132
statistics
from IRRUT200 about the database index 225
from IRRUT200 about the RACF database 230
how they affect system performance 29
on the RACF backup database 18, 38
STATISTICS option on SETROPTS 30
STOP command 78
stopping the RACF subsystem address space 78
storage requirement
coupling facility cache structure 91
RACF database 333
virtual for RACF 337
STORCLAS class
description 344
structure, cache
defining 90
name for 91
RACF support for REBUILDPERCENT 93
reconfiguring 93
size of 91
SUBSYSNM class
description 345

subsystem address space, RACF
stopping 78
subsystem, RACF
activating 70
assigning a user ID to 74
description 69
multiple 70
parameter library problems 327
recovery procedures 327
recovery when a task stops 328
restarting 76
restarting a function 76
running commands in 79
sample JCL to activate 75
shutting down 331
specifying the command prefix for 71
stopping 78
SURROGAT class
description 341
SWITCH operand
RVARY command 307
switchable RACF databases 8
switching primary and backup databases 305, 307
symptom records for SETROPTS processing on a sysplex 324
synchronization
of database profiles
establishing 145
maintaining 123
of passwords 122
when restoring a database 308
when using RVARY when the RACF database is shared 305
SYS1.LOGREC records for SETROPTS processing on a sysplex 324
SYS1.LPALIB
range table 45
SYS1.PARMLIB
IEFSSNxx member 71
SYS1.SAMPLIB
IRR@XACS member 188
RACF/DB2 external security module 188
SYSMVIEW class
description 341
SYSNAME keyword on TARGET command 157
sysplex communication
address space 89
cache structure 90
class descriptor table 49
command propagation 49, 322
coordinator 322
coupling facility 90
data sharing mode 88
effect of inactive backup data sets 90
enabling 89
failsoft mode 89
failsoft processing 22, 103
failures when propagating RVARY command 322
failures when propagating SETROPTS command 323
non–data sharing mode 88
sysplex communication *(continued)*
option 10
overview 84, 93
propagation of RVARY commands 103
re-IPLing a RACF data sharing group 40
read-only mode 89
RVARY SWITCH command 307
specifying in the data set name table 39, 89
sysplex data sharing
cache structure 90
coupling facility 90
coupling facility structure definition size 41
data sharing mode 88
defining structures in CFRM policy 90
failsoft processing 103
option 10
RACF support for REBUILDPERCENT 93
RACF support of rebuild interface 93
reconfiguring RACF structures 93
serialized access to shared databases 80
specifying in the data set name table 39, 89
sysplex recovery scenarios requiring non sysplex-communication/datasharing mode 315
sysplex recovery scenarios requiring XCF-local mode 314
SYSPRINT ddname
for IRRIRA00 utility 207
for IRRMIN00 utility 211
SYSRACF
for IRRMIN00 utility 211
system
main
configuring a new one 167
description 129
selecting 152
nonmain
description 129
System Display and Search Facility (SDSF)
See SDSF (System Display and Search Facility)
system generation
specifying bypass label processing 114
system name, identifying for a multisystem RRSF node 157
system operations
failures during operations on RACF-protected data sets 325
system performance
factors affecting 16
how exit routines affect 24
how failsoft processing can affect 22
how global access checking affects 24
how logging affects 20
how RACF commands can affect 21
how RACROUTE REQUEST=AUTH processing affects 32
how RACROUTE REQUEST=FASTAUTH processing affects 33
how RACROUTE REQUEST=VERIFY processing affects 31
how SETROPTS GENLIST processing affects 25
how SETROPTS RACLIST processing affects 25

system performance *(continued)*
how statistics gathering affects 29
how utility programs affect 22
using erase-on-scratch 23
using resident index and data blocks 19
system prerequisites 143
system programmer
role of 2
system utilities
RACF authorization checking during OPEN 107, 115

T

TABLE keyword
in PARM field of EXEC statement 238
IRRUT400 utility 239
tape labels
unlabeled tapes 115
tape volume protection
and bypass label processing 114
for unlabeled tapes 115
when moving tape volumes between systems 115
tape volume sets
conflicts detected by IRRUT400 utility 241
tape volumes
moving between systems 115
moving multivolume tape data sets between systems 115
using RACF to protect 114
using utilities when RACF-protected 115
TAPEVOL class
caution for the range table 46
description 341, 346
TARGET command 155
DELETE keyword 165
DESCRIPTION keyword 159
DORMANT keyword 165
LIST keyword 161
LOCAL keyword 159
LUNAME keyword 159
MAIN keyword 158
MODENAME keyword 160
NODE keyword 157
OPERATIVE keyword 164, 165
order in which to issue 174
PREFIX keyword 160
PROTOCOL keyword 159
PURGE keyword 165
SYSNAME keyword 157
TPNAME keyword 159
WORKSPACE keyword 160
target nodes
defining 156
tasks
(gerund phrase)
step for 6
configuring a multisystem node
steps for 179
synchronizing database templates
steps for 5, 7
TCBSENV pointer to ACEE 253
TCICSTRN class
description 342
TEMPDSN class
description 341
templates
determining the level in use 64
overview 5
RRSF considerations 143
updating 208
verification by IRRUT200 utility 228
temporary failsoft 103
TERMINAL class
description 341, 346
terminal monitor program
running IRRUT200 under TMP 222
TIMS class
description 343
Tivoli
general resource class 345
Tivoli Service Desk
general resource classes 344
TME 10 User Administration
requirement for RACF subsystem 70
TMEADMIN class
description 345
TMP (terminal monitor program)
running IRRUT200 under TMP 222
TPNAME keyword on TARGET command 159
TPUT 252
tracing APPC and IMAGE events 154
trusted attribute
for started procedures 95
TSO considerations
storing TSO profiles in the RACF database 106
using user ID in SYS1.UADS to logon 305
TSO information in the RACF database
operating considerations 106
TSO segment
using field level access checking to protect 106
TSO/E
general resource classes 345
TSOAUTH class
description 345
TSOPROC class
description 345
two-step method of password authentication 54, 275

U

UAUDIT operand
ALTUSER command
effect on system performance 20
UCBs above 16MB 114
UCICSTST class
description 342
UID mapping
improving performance 34
VLF class needed for 69
UIMS class
description 343

UL tapes 115
undefined user
  supplying a user ID 298
unit control blocks (UCBs) above 16MB 114
universal group, effect on performance 35
UNIXMAP class
  description 345
  how it improves performance 34
UNIXPRIV class
  description 345
unlabeled tapes 115
unlocking a database
  example using IRRUT400 244
UNLOCKINPUT keyword
  IRRUT400 utility 238
user data set
  preventing accidental destruction of data 283
user ID
  assigning to RACF subsystem 74
  listing all occurrences on the RACF database 212
  protected 74, 94
  used by RACF 2
user ID association
  overview 120
user private, below 16MB
  storage requirement 338
user security packet 34, 69
users
  information on provided by IRRUT100 utility 213
USP 34, 69
utilities
  database cross reference 212
  database initialization 208
  database reorganization 203
  database split/merge/extend 232
  database unload 245
  database verification 218
  for use on the RACF database 202
  how they can affect system performance 22
  IRRADU00 245
  IRRBRW00 246
  IRRDBU00 245
  IRRIRA00 203
  IRRMIN00 208
  IRRRID00 245
  IRRUT100 212
  IRRUT200 218
  IRRUT400 232
  RACF report writer 245
  RACFICE reporting tool 246
  RACFRW 245
  remove ID 245
  RRSF VSAM file browser 246
  SMF data unload 245
  using IEHMOVE with the ADSP attribute 109
  using IEHMOVE with the COPYAUTH parameter 110
  using on RACF-protected DASD data sets 107
  using on RACF-protected tape volumes 115
  using with the group-OPERATIONS attribute 108
  using with the OPERATIONS attribute 108
utility control statements
  for IRRUT100 utility 215
  for IRRUT200 utility 223

V

VCICSCMD class
  description 342
verifying RACF users 2
virtual storage requirement for RACF 337
VLF
  GID mapping, class needed for 69
  IRRACEE class 32, 67
  IRRGMAP class, defining 69
  IRRGTS class 80
  IRRSMAP class, defining 69
  IRRUMAP class, defining 69
  removing information from 68
  UID mapping, class needed for 69
  using to improve performance 32, 34
VMBATCH class
  description 346
VMBR class
  description 346
VMCMD class
  description 346
VMEVENT class
  description 346
VMMAC class
  description 346
VMMDISK class
  description 346
VMNODE class
  description 347
VMPOSIX class
  description 347
VMRDR class
  description 347
VMSEGMT class
  description 347
VMXEVENT class
  description 347
VSAM data set
  failures during ALTER operation 326
  failures during DEFINE operation 326
  failures during DELETE operation 326
  IMPORT command 113
  IMPORTRA command 113
  LISTCAT command 113
  moving multivolume data sets between systems 112
  REPRO command 113
  RESETCAT command 113
  RRSF VSAM file browser 246
  turning off the RACF indicator and deleting the profile 111
  using access method service commands 113
VTAM (Virtual Telecommunications Access Method)
  general resource class 341
VTAM considerations for an RRSF network 145

VTAMAPPL class
  description 341
VXMBR class
  description 347

W

WDSQUAL keyword on TARGET command 134, 135, 158, 160
WIMS class
  description 343
work data set for IRRUT100
  format of the records 214
worksheet, RRSF 349
workspace data sets
  defining 135, 160
  description 134
  determining how full 136
  increasing the size of 329
  maintaining 136
  preallocating 135
  prefix for 160
  purging 165
  recovering from VSAM errors on 329
  recovering when they fill up 329
  RRSF VSAM file browser 246
  size guidelines 136
WORKSPACE keyword on TARGET command 160
WRITER class
  description 341, 347
WTO 252

X

XAPLFUNC function codes 196

Z

z/OS UNIX
  general resource classes 345
z/OS UNIX System Services
  application performance 34, 69

Readers' Comments — We'd Like to Hear from You

z/OS
Security Server RACF System Programmer's Guide

Publication No. SA22-7681-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

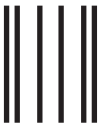
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SA22-7681-04

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5694-A01, 5655–G52

Printed in USA

SA22-7681-04