z/OS
Version 2 Release 4

Security Server RACF System Programmer's Guide

SA23-2287-40

Note

Before using this information and the product it supports, read the information in “Notices” on page 367.

This edition applies to Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2020-06-23

© Copyright International Business Machines Corporation 1994, 2020.
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
RACF courses xv
Other sources of information xv
Internet sources xvi

How to send your comments to IBM ..... **xvii**
If you have a technical problem xvii

Summary of changes ..... **xix**
Summary of changes for z/OS Version 2 Release 4 (V2R4) xix
Summary of changes for z/OS Version 2 Release 3 (V2R3) xix
Summary of changes for z/OS Version 2 Release 2 (V2R2) as updated December 2016 xx
Summary of changes for z/OS Version 2 Release 2 (V2R2) xx

Chapter 1. Security and the RACF database **1**
Data processing security 1
How RACF meets security needs 1
Identifying and verifying users 1
Authorizing users to access resources 1
Controlling access to resources 1
Logging and reporting 2
Administering security 2
Basic RACF concepts 2
RACF and the operating system 2
The RACF database 3
Database templates 3
Multiple data set support 6
Backup RACF database 6
Shared RACF databases 7
Sharing RACF data without sharing a database 9
Creating a RACF database 10
Finding a location for the RACF database 10
Copying your database 11
Using DFSMSdss DEFRAG 12
DFSMS enhanced data integrity (EDI) 12
Monitoring the usable space in your RACF database 12

Chapter 2. Performance considerations **15**
The RACF database 15
Selection of control unit and device 15
Shared RACF database 15
Multiple data sets 16
Database housekeeping 16

Creating backup RACF databases....16
Resident data blocks....17
RVARY SWITCH command....18
Auditing....18
Operands requiring the AUDITOR attribute....18
RACF commands....19
RACF utility programs....20
BLKUPD....20
IRRUT200....20
Failsoft processing....20
Protecting data during its life cycle....21
Encryption of data at rest....21
Erase-on-scratch....22
Installation-written exit routines....22
Using global access checking....22
The SETROPTS command....23
Using SETROPTS RACLIST and SETROPTS GENLIST....23
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....32
Program signing and signature verification....33
Using generic profiles....33
Customizing the number of lists of generic profiles that RACF maintains....34
Mapping UIDs to user IDs and GIDs to group names....35
z/OS UNIX System Services applications....35
Large profiles....35
Large groups....36
Universal groups....36

Chapter 3. RACF customization....37
RACF database configuration....37
Overview....37
Specifying RACF database options....40
Other considerations....49
Specifying resource-class options....49
The class descriptor table (CDT)....50
The RACF router table....53
ENF signals....54
Type 62 ENF signals....54
Type 71 ENF signals....55
Type 79 ENF signals....55
Password authentication options....56
The RACF encryption algorithms (KDFAES and DES)....56
Using the masking algorithm....58
Using your own authentication algorithm....58
PassTicket authentication....58
Planning Considerations for enabling KDFAES....59
Changing the RACF report writer options (ICHRSMFI module)....62
Customizing the RACF remote sharing facility....65

Chapter 4. Operating considerations....67
Enabling and disabling RACF....67
Enabling RACF....67
Disabling RACF....67
Dynamic parse and IRRDPI00....67

Syntax of the IRRDPI00 command....68
IRRDPI00 errors and return codes....71
RACF authorization of the IRRDPI00 command....71
TSO/E authorization of the IRRDPI00 command....71
Automating IRRDPI00....72
ACEEs and VLF considerations....73
Dependencies....73
Operation....74
Removing information from VLF....74
VLF considerations for mapping UIDs and GIDs....74
Dependencies....75
VLF considerations for caching user security packets (USPs)....75
Dependencies....75
VLF considerations for program signature verification....75
Dependencies....75
The RACF subsystem....75
Activating the RACF subsystem....77
Restarting the RACF subsystem....82
Restarting a function in the RACF subsystem....83
Stopping the RACF subsystem address space....85
Diagnosing problems in the RACF subsystem....86
RACF operator commands....86
Group tree in storage....86
Shared database considerations....87
Using the global resource serialization function....87
RACF ENQ resources....88
Sysplex considerations....91
Sharing a database....92
Sysplex communication....100
Enabling sysplex communication....101
System authorization facility (SAF)....105
The SAF router....105
The SAF callable services router....105
Associating started procedures and jobs with user IDs....105
Methods for associating started procedures with RACF identities....107
The STARTED class....108
The started procedures table (ICHRIN03)....109
Coding the started procedures module....109
The ICHAUTAB module....112
Failsoft processing....112
General considerations....113
Impact on users....114
CICS considerations....114
CICS timeout value....114
TXSeries....115
DFSMS considerations....115
TSO considerations....115
ISPF considerations....115
Db2 considerations....116
DASD data sets....116
Using utilities on RACF-protected DASD data sets....116
Moving a RACF-indicated DASD data set between systems....119
Using access method services commands....121
DASD volumes....122
Scratching DASD data sets....122
Moving DASD volumes between systems....122
UCBs above 16MB....122
Protecting tape data....122

Tape data protection and bypass label processing (BLP)....122
Considerations for unlabeled (NL) tapes....123
Using utilities on RACF-protected tape volumes and tape data sets....123
Moving tape volumes between systems....123
Moving multivolume tape data sets between systems....123
Multiple users per address space....123
Restarting jobs....124
Panel driver interface....124
REXX RACVAR function....124
Installing the REXX RACVAR function....124
Using the REXX RACVAR function....125
Initializing RACF verification of signed programs (IRRVERLD)....125
IRRVERLD return codes....125

Chapter 5. RACF remote sharing facility (RRSF)....127
Overview of the RACF remote sharing facility (RRSF)....127
Understanding the RRSF concepts....127
Overview of the RRSF function....127
The RRSF network....128
RRSF nodes....129
Connections between nodes....132
Network protocols....134
Workspace data sets....136
How a directed command travels through the network....140
Defining an RRSF environment....143
Preparing to configure an RRSF network....145
Configuring an RRSF network....154
The RACF parameter library....184
Customizing and establishing security for RRSF....191
Examples of defining a remote sharing environment....194
Monitoring your remote sharing environment....200

Chapter 6. RACF database utilities....203
RACF internal reorganization of aliases utility program (IRRIRA00)....204
IRRIRA00 stage conversion....206
Diagnostic capability....207
Input for IRRIRA00....208
Output from IRRIRA00....208
RACF database initialization utility program (IRRMIN00)....209
Running IRRMIN00 when PARM=NEW is specified....210
Running IRRMIN00 when PARM=UPDATE is specified....211
Running IRRMIN00 when PARM=ACTIVATE is specified....211
Diagnostic capability....212
Input for IRRMIN00....212
Output from IRRMIN00....212
RACF cross reference utility program (IRRUT100)....213
Group name and user ID occurrences that IRRUT100 lists....213
Exit routine....214
Diagnostic capability....214
The work data set....215
Using IRRUT100....216
RACF database verification utility program (IRRUT200)....218
Copying a data set in the RACF database....219
Diagnostic capability....220
Monitoring the capacity of the RACF database....220
Processing considerations for databases from other systems....221
Using IRRUT200....221

Utility control statements..... 225
Scanning the index blocks..... 226
BAM/allocation comparison.....229
IRRUT200 return codes..... 232
RACF database split/merge/extend utility program (IRRUT400)..... 233
How IRRUT400 works.....233
Using IRRUT400 to extend a database..... 234
Copying a RACF database..... 234
Repairing a RACF database.....235
Diagnostic capability..... 236
Executing IRRUT400.....236
IRRUT400 return codes..... 241
IRRUT400 examples..... 242
Utilities documented in other documents..... 244
RACF database unload utility program (IRRDBU00)..... 244
RACF remove ID utility (IRRRID00).....244
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)..... 245
RACFICE reporting tool..... 245

Chapter 7. RACF installation exits..... 247
Overview..... 247
RACF exits report..... 248
Extended addressing for exits..... 248
Data set naming convention table..... 249
Exits running in the RACF subsystem address space..... 250
Possible uses of RACF exits..... 250
Summary of installation-exit callers.....250
ACEE compression/expansion exits.....253
Range tables.....254
IRRACX01.....255
IRRACX02 257
Command exits for specific commands.....259
ICHCNX00 processing..... 259
ICHCCX00 processing.....262
Common command exit..... 263
Controlling the exit routine through the dynamic exits facility.....263
Replacing the exit routine..... 264
Exit routine environment..... 264
Exit recovery.....264
Exit routine processing..... 264
Programming considerations.....267
Entry specifications.....267
Return specifications..... 267
Coded example of the exit routine..... 268
New-password exit.....268
ICHPWX01 processing.....269
Using the exit for password quality control.....271
Coded example of the exit routine..... 271
New-password-phrase exit (ICHPWX11)..... 272
Installing the exit routine.....272
Exit routine environment..... 273
Exit routine processing..... 273
Programming considerations.....274

Entry specifications....274
Return specifications....275
Coded example of the exit routine....276
Password authentication exits....276
ICHDEX01....277
ICHDEX11....279
RACROUTE REQUEST=AUTH exits....281
Extended addressing....281
Preprocessing exit (ICHRCX01)....281
Postprocessing exit (ICHRCX02)....282
Possible uses of the exits....283
RACROUTE REQUEST=DEFINE exits....284
Extended addressing....285
Automatic direction of application updates....285
Preprocessing exit (ICHRDX01)....285
Postprocessing exit (ICHRDX02)....286
RACROUTE REQUEST=FASTAUTH exits....286
Preprocessing exits (ICHRFX01 and ICHRFX03)....287
Postprocessing exits (ICHRFX02 and ICHRFX04)....290
Possible uses of the exits....296
RACROUTE REQUEST=LIST exits....297
Pre- and postprocessing exit (ICHRLX01)....298
Selection exit (ICHRLX02)....298
RACROUTE REQUEST=VERIFY(X) exits....299
Preprocessing exit (ICHRIX01)....300
Postprocessing exit (ICHRIX02)....301
RACF report-writer exit....302
ICHRSMFE processing....303
Custom Field Validation Exits....303
Custom Field Validation Exit (IRRVAF01)....303
Custom Field Validation Exit (VALREXX)....306
SAF router exits....308

Chapter 8. Recovery procedures....309
Overview....309
Exit routine considerations....309
TSO considerations....309
The RVARY command....309
Synchronization considerations....312
Considerations for issuing RVARY from the RACFRCVY started procedure....312
Failures on the RACF database....313
Sample recovery procedures....313
Failures using sysplex data sharing....315
Read-only mode....315
Non–data sharing mode....316
Recovery scenarios....316
Failures during RACF command processing....319
Commands that do not modify user-created RACF profiles....319
Commands that have recovery routines....320
Commands that perform single operations....321
Commands that perform multiple operations....321
Recovering from errors in identity mapping profiles....323
Recovering from errors with application identity mapping....324
Commands that are propagated for RACF sysplex communication....325
Failures during RACF manager processing....327
Failures during system operations on RACF-protected data sets....328
Failures during SCRATCH or DELETE....328

Failures during ALLOCATE or DEFINE..... 328
Failures during RENAME or ALTER..... 329
Failures during EOV (non-VSAM)..... 329
Failures in the RACF subsystem address space..... 329
Recovering from RACF parameter library problems..... 329
Recovering when a task stops..... 330
Recycling an RRSF connection..... 330
Recovering from VSAM errors on the RRSF workspace data sets..... 331
The last resort—shutting down the RACF subsystem address space..... 333

Chapter 9. Storage estimates..... 335
RACF database storage requirements..... 335
Factors affecting the size of the RACF database..... 335
Formula for the RACF database size..... 335
RACF virtual storage requirements..... 339
Coupling facility cache structure storage requirements..... 342

Appendix A. RRSF initialization worksheet and scenario..... 343
RRSF node configuration worksheet..... 343
RRSF initialization scenario..... 344
Background information..... 344
Completed RRSF node configuration worksheet for node MVS01..... 345
Completed RRSF node configuration worksheet for node MVS02..... 347
Summary..... 348
Detailed instructions..... 349
Now it's your turn to fill out the worksheet..... 354

Appendix B. Non-recommended options..... 355
Selecting options with ICHSECOP..... 355
Bypassing RACF initialization processing..... 355
Selecting the number of resident data blocks..... 356
Disallowing duplicate names for data set profiles..... 356
Changing the ICHAUTAB module..... 356
Using the RACF authorized-caller table..... 356

Appendix C. MAIN switch procedure in a mixed-release network..... 359
Old or new MAIN is downlevel..... 359
Old and new MAIN are uplevel..... 360

Appendix D. Accessibility..... 363
Accessibility features..... 363
Consult assistive technologies..... 363
Keyboard navigation of the user interface..... 363
Dotted decimal syntax diagrams..... 363

Notices..... 367
Terms and conditions for product documentation..... 368
IBM Online Privacy Statement..... 369
Policy for unsupported hardware..... 369
Minimum supported hardware..... 369
RSA Secure code..... 370
Programming Interface Information..... 370
Trademarks..... 370

Index..... 371

Figures

1. RACF and its relationship to the operating system..... 3

2. ICHRDSNT example 1 — for three data sets..... 46

3. ICHRDSNT example 2 — data sharing option and split database.....47

4. ICHRRNG example for three data sets..... 48

5. Setting up the RACF subsystem address space..... 78

6. Coding ICHRIN03 so the assembler calculates the count field..... 110

7. An RRSF network..... 129

8. An RRSF network containing a single-system node and a multisystem node..... 130

9. An RRSF network with two multisystem nodes and one single-system node..... 131

10. A directed command traveling through an RRSF network that uses APPC..... 141

11. Sample output from the SET LIST command.....156

12. Summary information displayed by a TARGET LIST command for a single-system node..... 164

13. Detailed information displayed by a TARGET LIST command for a single-system node using APPC/MVS..... 165

14. Detailed information displayed by a TARGET LIST command for a single-system node using TCP/IP..... 166

15. Detailed information displayed by a TARGET LIST NODE(nodename) command for a single-system local node that uses both APPC/MVS and TCP/IP..... 166

16. Information displayed by a TARGET LISTPROTOCOL command..... 167

17. Information displayed by a TARGET LIST command..... 167

18. Summary information from the TARGET LIST command for a multisystem node..... 167

19. Detailed information from the TARGET LIST command for a system on a multisystem node..... 168

20. Summary information from the TARGET LIST command for a multisystem node that uses TCP/IP...168

21. Detailed information from the TARGET LIST command for a system on a multisystem node..... 169

22. Example from the TARGET LIST command for local nodes that use IPv6, no address specified..... 170

23. Detailed information from the TARGET LIST command output for a remote TCP/IP node when TCP/IP IPv6 is enabled, host name is specified..... 171

24. Detailed information from the TARGET LIST command output for a remote TCP/IP node when TCP/IP IPv6 is enabled and an IPv4 address is specified..... 172

25. TARGET LIST output showing workspace data sets for both the old and new protocols during a protocol conversion..... 183

26. Example of a RACF parameter library for a node running in local mode..... 188

27. Example of a RACF parameter library for a node running in remote mode..... 190

28. Two RRSF nodes in local mode..... 194

29. An RRSF network with two nodes..... 195

30. An RRSF network containing a multisystem node and a single-system node.....197

31. An RRSF network containing two multisystem nodes.....198

32. Sample output from IRRUT100.....218

33. Sample output of formatted index produced by IRRUT200 (part 1 of 3)..... 227

34. Sample output of formatted index produced by IRRUT200 (part 2 of 3)..... 228

35. Sample output of formatted index produced by IRRUT200 (part 3 of 3)..... 228

36. Sample output of formatted alias index produced by IRRUT200..... 229

37. Sample output of encoded BAM map produced by IRRUT200.....232

38. Logic that determines whether ICHRFX02 or ICHRFX04 is called..... 291

39. The number of blocks that are required for the profiles.....336

40. The number of index blocks required..... 338

41. The number of blocks that are required for the alias index..... 339

42. The number of BAM blocks required.....339

43. RACF storage use.....340

Tables

1. Mechanisms intended to address threats to data during its life cycle..... 21

2. Alias index entry values..... 39

3. ENF 62 event code..... 55

4. ENF 71 event code..... 55

5. ENF 79 event code..... 56

6. Format of ICHRSMFI..... 63

7. Valid profile types and segment names for IRRDPI00..... 70

8. Functions you can restart using the RESTART command 83

9. RACF ENQ resources..... 88

10. Potential data corruption messages displayed at IPL time..... 95

11. Advantages of creating IRRPLEX_sysplex-name profiles manually..... 97

12. Potential data corruption messages displayed when issuing the RVARY ACTIVE command..... 98

13. Connection states between nodes..... 133

14. Example of workspace data set names..... 138

15. Defining an RRSF environment—summary of tasks..... 143

16. RRSFDATA resource names..... 192

17. RACF utilities described in this chapter..... 203

18. IRRIRA00 stage summary..... 206

19. RACF installation-exits cross-reference table—Part 1 of 2..... 250

20. RACF installation-exits cross-reference table—Part 2 of 2..... 252

21. ICHCNX00-exit parameter processing..... 260

22. Fields available during ICHPWX01 processing..... 270

23. Availability of parameters during ICHPWX11 processing for each RACF component that can invoke the exit 275

24. REXX input values 307

25. RACF estimated storage usage 341

26. RRSF node configuration worksheet 343

27. RRSF node status worksheet 344

About this document

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 RACF to satisfy an installation's particular needs

You should be familiar with the information in:

- *z/OS Security Server RACF Security Administrator's Guide*
- *z/OS Upgrade Workflow*
- 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® or TCP/IP.

Where to find more information

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS library, including the z/OS Knowledge Center, see the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

To find educational material, see the IBM Education home page (www.ibm.com/services/learning).

RACF courses

The following RACF classroom courses are available in the United States:

ES191
Basics of z/OS RACF Administration

BE870
Effective RACF Administration

ES885
Exploiting the Advanced Features of RACF

IBM® provides various 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)

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.

© Copyright IBM Corp. 1994, 2020

Internet sources

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

- z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary)
- IBM Redbooks (www.ibm.com/redbooks)
- Enterprise security (www.ibm.com/systems/z/solutions/enterprise-security.html)
- RACF home page (www.ibm.com/us-en/marketplace/resource-access-control-facility-racf)
- RACF download page (github.com/IBM/IBM-Z-zOS/tree/master/zOS-RACF/Downloads)

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead “If you have a technical problem” on page xvii.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS Security Server RACF System Programmer's Guide, SA23-2287-40
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

© Copyright IBM Corp. 1994, 2020

Summary of changes

Summary of changes for z/OS Version 2 Release 4 (V2R4)

The following changes are made to z/OS Version 2 Release 4 (V2R4).

New

June 2020 refresh

- Enhancements were made clarifying user ID authority. See “Assigning a user ID to the RACF subsystem” on page 80 and “Commands that can be issued from the RACF parameter library” on page 185 for more information.

Prior to June 2020 refresh

- APAR OA56715 updates the impact of using the DFSORT ICEGENER utility on the RACF data base verification utility program (IRRUT200). See: “RACF database verification utility program (IRRUT200)” on page 218.
- The RACF installation-exits cross-reference table has been updated to include an entry in the IRRVAF01 column for ADDSD, ALTDSD, RALTER, and RDEFINE commands. See “Summary of installation-exit callers” on page 250 for more information.
- Exit IRRVAF01 has been updated to include the RDEFINE and RALTER commands and the ADDSD and ALTDSD commands. See “Custom Field Validation Exit (IRRVAF01)” on page 303 for more information.
- A new topic has been added for the VALREXX interface. See “Custom Field Validation Exit (VALREXX)” on page 306 for more information.
- A new topic has been added for maintaining RRSF workspace data sets. See “Modifying the attributes of workspace data sets” on page 139 for more information.
- A couple of statements regarding authorization in RACROUTE REQUEST=VERIFY have been updated to include the new IDTA keyword. See “Changing the ICHAUTAB module” on page 356 and “Format of the authorized-caller table” on page 357.

Changed

Prior to June 2020 refresh

- “PassTicket authentication” on page 58 has been updated to provide more clarity and to provide links to relevant information.

Deleted

Prior to June 2020 refresh

- The topic "How RACF processes the password or PassTicket" has been removed. Please refer to How RACF processes the PassTicket in *z/OS Security Server RACF Security Administrator's Guide*.

Summary of changes for z/OS Version 2 Release 3 (V2R3)

The following changes are made to z/OS Version 2 Release 3 (V2R3).

New

- The following updates are for APAR OA52650:
  - A new method for RACF database configuration is introduced, “IRRPRMxx parmlib member” on page 40.

© Copyright IBM Corp. 1994, 2020

  - The following IRRUT400 utility topics are updated to include support for the IRRPRMxx parmlib member:
    - “Executing IRRUT400” on page 236
    - “Selecting the output data set” on page 238
    - “Specifying parameters” on page 238
    - “Example 2a. Splitting a database using the RANGE keyword” on page 242
    - “Example 2b. Splitting a database” on page 243
- “DFSMS considerations” on page 115 and “Encryption of data at rest” on page 21have been updated to include information on the new DFP field, DATAKEY.

Changed

- “IRRPRMXX parmlib sample” on page 44 is updated with the corrected sample name.
- The considerations for DFSMS have been updated. See “DFSMS considerations” on page 115 for more information.

Summary of changes for z/OS Version 2 Release 2 (V2R2) as updated December 2016

The following changes have been made in z/OS Version 2 Release 2 (V2R2) as updated December 2016.

New information

- New information about WHEN(PROGRAM) and NOWHEN(PROGRAM) has been added to “RACF authorization of the IRRDPI00 command” on page 71.

Changed information

There is no changed information in this version.

Deleted information

There is no deleted information in this version.

Summary of changes for z/OS Version 2 Release 2 (V2R2)

The following changes have been made in z/OS Version 2 Release 2 (V2R2).

New information

- “RACF internal reorganization of aliases utility program (IRRIRA00)” on page 204 has been updated with new information.
- The SIGNAL function has been added to Table 8 on page 83 in “Restarting a function in the RACF subsystem” on page 83.
- A new MAINSWITCHINPROGRESS ENQ has been added to Table 9 on page 88.
- A new topic “Allocating workspace files for remote non-MAIN connections in preparation for dynamic main switches” on page 157 has been added to “The SET command” on page 155.
- The topic **Configuring a new main system in a multisystem node** has been replaced by “Configuring a new main system in a multisystem node” on page 175.

- ICHDEX01 in "Password authentication exits" on page 276 has had the following topics updated: "Exit routine environment" on page 277, "Exit routine processing" on page 277, and "Programming considerations" on page 277
- A new Redbook about RACF Remote Sharing Facility has been listed in "Monitoring your remote sharing environment" on page 200.
- Information about the IRRXUTIL interface has been added to "Monitoring your remote sharing environment" on page 200.
- New information has been added to the steps listed in "Configuring two multisystem nodes" on page 198 to accommodate for dynamic MAIN switching.
- New information was added to the following topics in support of enhanced RACF password security:
  - "Factors affecting the size of the RACF database" on page 335
  - "Password authentication exits" on page 276
  - "New-password-phrase exit (ICHPWX11)" on page 272
  - "ICHPWX01 processing" on page 269
  - "Password authentication options" on page 56

Changed information

- "RACROUTE REQUEST=VERIFY(X) exits" on page 299 and "Expansion-time invocation of IRRACX01" on page 256 are updated with information as to when the exits will get control.
- "Controlling outgoing requests from the local node" on page 172 was updated with a note about the DENYINBOUND keyword of the **TARGET** command.
- "Defining RRSF nodes to RACF using the TARGET command" on page 159 has been updated with the DENYINBOUND, NEWMAIN, PLEXNEWMAIN, and RESETDENYINBOUNDCOUNT keywords.
- "Defining RRSF nodes to RACF using the TARGET command" on page 159 was updated with information regarding the DENYINBOUND and ALLOWINBOUND keywords of the **TARGET** command.
- The list of actions "The SET command" on page 155 allows you to perform has been updated.
- **Defining RRSF nodes to RACF** in "The SET command" on page 155 has been retitled "Defining RRSF nodes to RACF using the TARGET command" on page 159
- "Initializing RACF verification of signed programs (IRRVERLD)" on page 125 has been updated with a note about systems that share a RACF database.
- Figure 11 on page 156 in "Listing information about RRSF functions on the local node" on page 155 has been updated with new information.
- The information previously in the topic **Configuring a new main system in a multisystem node** has been restructured and is now located in Appendix C, "MAIN switch procedure in a mixed-release network," on page 359.
- "RACF cross reference utility program (IRRUT100)" on page 213 is updated for the ROAUDIT attribute.
- "RACF exits report" on page 248 is updated to indicate that the AUDITOR or ROAUDIT attribute is required to run the report.
- "RACROUTE REQUEST=FASTAUTH exits" on page 286, "Preprocessing exits (ICHRFX01 and ICHRFX03)" on page 287and "Postprocessing exits (ICHRFX02 and ICHRFX04)" on page 290 have been updated for FSEXEC class.
- "Erase-on-scratch" on page 22 has been updated for clarity.
- "Copying a data set in the RACF database" on page 219 has been updated to clarify that the in-use backup RACF database is an inactive database.
- The topic Planning considerations has been renamed to "Planning Considerations for enabling KDFAES" on page 59.

Deleted information

- References to unsupported releases of z/Os have been removed in Chapter 6, “RACF database utilities,” on page 203.
- Information that is no longer relevant, regarding when ICHPWX01 is invoked through the ALTUSER command, was deleted from “ICHPWX01 processing” on page 269.

Chapter 1. Security and the RACF database

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.

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 can access resources, but also the way the user can 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.

© Copyright IBM Corp. 1994, 2020

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.

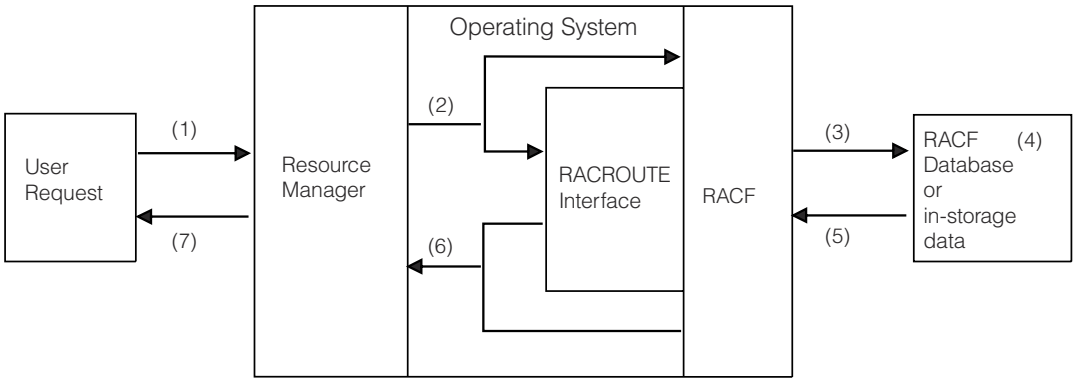
RACF and the operating system

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 3 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 335.

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 information about RRSF functions on the local node" on page 155.

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 209.

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

About this task

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" on page 5.

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

Procedure

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.

Results

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

About this task

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.

Procedure

Run IRRMIN00 specifying PARM=UPDATE.

Results

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

About this task

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 must have installed the PTF.

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

Procedure

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 ACTRDS section of SYS1.SAMPLIB member RACJCL illustrates how to invoke IRRMIN00 specifying PARM=ACTIVATE.

   ———————————————————————————————————————

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

   You must ensure that the dynamic parse data set and the in-storage templates are at the same level. Most installations invoke IRRDPI00 automatically during an IPL to accomplish that. Because you do not need to re-IPL, you must invoke IRRDPI00 yourself. The ACTRDS section of SYS1.SAMPLIB member RACJCL illustrates how to invoke IRRDPI00 specifying UPDATE. For more information about dynamic parse, see “Dynamic parse and IRRDPI00” on page 67.

   ———————————————————————————————————————

Results

When you are done, both the in-storage templates and those on the database are the latest level, the level that is 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 to control the amount of updating to the backup database. For information about data set name tables, 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 an I/O error on a data set in the primary database, resulting in the device to be varied offline or boxed, RACF automatically switches to your active backup RACF data set without requiring the operator to enter a password. 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 313.

For additional information on backing up your database, see “Creating backup RACF databases” on page 16.

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 system not configured as a RRSF node, including a system running z/VM®. Be aware that if the RRSF node is using password synchronization or automatic direction, database changes made from a system that is not an RRSF node are not propagated to other RRSF nodes, and database inconsistencies can result. See Chapter 5, “RACF remote sharing facility (RRSF),” on page 127 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*.

Tip: To determine whether the database is on a device that has been configured as shared, issue an RVARY LIST command. If the device is not shared, the output includes a column labeled SHR, with the value N. The column does not appear if the device is shared.

The RACF database templates must match the latest level of code on the sharing systems. Use the IRRMIN00 utility to update the database templates when you install a new release or service level. Because the database structure changed for z/OS V1R8 to allow database templates that are larger than one 4K block, the z/OS V1R8 database templates ares not downwardly compatible unless you install an APAR on the lower-level system.

Requirement: To share a database between a system running z/OS V1R8 (or higher) and a system running a lower-level release of z/OS, you must install APAR OA12443 on the lower release. The APAR is available for z/OS V1R4, V1R5, V1R6, and V1R7. An APAR is not required on z/VM. You can share a RACF database between a system running z/OS V1R8 (or higher) and a z/VM system.

Restriction: If you are sharing a RACF database between a system running z/OS V1R8 (or higher) and a z/OS V1R4 system, do not run the following utilities from the z/OS V1R4 system. Run them only from a system running z/OS V1R8 (or higher) or run them from a z/OS V1R5, V1R6, or V1R7 system with APAR OA12443 installed:

- IRRMIN00
- IRRUT200
- IRRUT400
- IRRUT300 (BLKUPD)
- IRRDBU00

- IRRIRA00

Attention: If you have run the IRRIRA00 utility to convert the RACF database to application identity mapping stage 1 or later, note the following:

- You should not run the IRRUT400 utility from a downlevel system.
- All systems that update the OMVS segment of USER or GROUP profiles, or update the ALIAS segment of general resource profiles (for example, any SERVAUTH class profile), or run RACF utilities, should have global resource serialization connections between the systems, should be in the same global resource serialization complex, and should be running OS/390® release 10 or any z/OS release. Adding or deleting a profile that has any of these segments, altering these segments, or running RACF utilities from a system outside the global resource serialization complex might result in incorrect results; for example, an alias index entry for an OMVS UID or SERVAUTH alias might point to the wrong profile, or to one that does not exist. To prevent database sharing errors, it might be useful to use RACF program control to restrict access to all RACF commands that can update these segments, to ensure that they cannot be used from systems outside a single global resource serialization complex.

  If you do get your alias index out of synchronization with the USER or general resource profiles, you might need to delete and re-create some profiles or alter some data (for example, a UID or GID), in order to correct the inconsistency. For more information, see "Recovering from errors with application identity mapping" on page 324.

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. For more information on password authentication, see "Password authentication options" on page 56.

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

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

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

Considerations when sharing between z/OS and z/VM systems

Restrictions: When you share a RACF database between z/OS and z/VM systems, the following restrictions apply:

- The RACF database cannot be on FBA DASD.
- You cannot use a coupling facility for the RACF database on the z/OS system.
- You must use RESERVE/RELEASE serialization for the database. You cannot use the MVS global resource serialization function (or an equivalent product) on the z/OS system to convert the RESERVEs to ENQs.
- You must perform administration of many profiles from the z/OS system. For example, if a USER profile contains alias mapping fields (for example, OMVS UID), those users should be managed from the z/OS side so that the indexes are properly maintained. In the OMVS UID example, there is no OMVS keyword on VM, so you could not directly manage the segment on VM. However, if you deleted the profile on the VM system, the alias indexes would not be properly maintained on the z/OS system. In a similar example, if a USER has digital certificates on z/OS, and that user were deleted on the VM side, the digital certificates would not be cleaned up properly on z/OS.

Guideline: If you are sharing a database between z/VM and z/OS systems, run the utilities from the z/OS side for better ease-of-use, recovery, and error-reporting.

In a remote sharing environment, a z/OS system configured as an RRSF node can share a RACF database with a z/VM system. Database updates that are made on other RRSF nodes can be propagated to the shared database, allowing the z/VM system to share database changes made on other systems. However, database updates that are made on the z/VM system are not propagated to the RRSF nodes.

Considerations for the RACGLIST class

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 global resource serialization complex. The major name SYSZRAC2 cannot be in the exclusion resource name list (RNL).

Considerations for classes that do not allow generic profile processing

Beginning with z/OS V1R8, you can specify that a class does not allow generic profile processing, using the GENERIC=DISALLOWED keyword on the ICHERCDE macro, or the CDTINFO(GENERIC(DISALLOWED)) keyword on the RDEFINE or RALTER command. If you are sharing a z/OS V1R8 (or higher) RACF database with a lower-level system, and you plan to define classes that do not allow generic profile processing, there are some things that you need to consider:

Considerations for dynamic classes

- You must administer the profile in the CDT class that contains the GENERIC(DISALLOWED) keyword from the system running z/OS V1R8 or higher. If you define a CDT profile containing the GENERIC(DISALLOWED) keyword on a system running z/OS V1R8 (or higher) and then change that profile from a lower-level system that shares the same RACF database, it is possible that the dynamic class defined by the profile will become unusable on the z/OS V1R8 (or higher) system.
- You must administer the profiles in the class defined using the GENERIC(DISALLOWED) keyword from the system running z/OS V1R8 or higher. If you administer profiles in the class from a lower-level system that shares the same RACF database, you might be able to activate generic profile processing on the lower-level system because the GENERIC(DISALLOWED) keyword is not recognized on that system. Then you can add generic profiles from either system.

Considerations for static classes

- If the installation class descriptor table (ICHRRCDE) contains the GENERIC=ALLOWED or GENERIC=DISALLOWED keyword, you must administer it from the from the system running z/OS V1R8 or higher. Once the installation class descriptor table is assembled on the system running z/OS V1R8 or higher, the object code for ICHRRCDE can be linked and used on the lower-level systems.
- You must administer the profiles in the class that was defined using the GENERIC=DISALLOWED keyword from the system running z/OS V1R8 or higher. If you administer the profiles from a lower-level system, you might be able to activate generic profile processing on that lower-level system because GENERIC=DISALLOWED is not recognized on that system. Then you can add generic profiles from either system.

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 a coupling facility is available. RACF data sharing might improve data access performance.

For more information on these options, see "Sysplex considerations" on page 91.

Sharing RACF data without sharing a database

Installations often find it 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 by way of VTAM and APPC/MVS or TCP/IP, 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 about the RACF remote sharing facility, see Chapter 5, "RACF remote sharing facility (RRSF)," on page 127.

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 about using IRRMIN00, see "RACF database initialization utility program (IRRMIN00)" on page 209. A RACF database must be allocated in one extent.

Guideline: 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 guarantees 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 that 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, by way of control statements.

See "RACF database storage requirements" on page 335 for information about estimating the amount of space a database requires.

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" on page 11.

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.

Tip: When you issue the RVARY commands shown in the following sections, use the DATASET operand to name the data sets that you are processing, 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" on page 11, 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 might 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" on page 11 and "7" on page 11 where your databases might 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 "3.a" on page 11 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 "3.b" on page 12 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 without PARM=ACTIVATE, or IRRUT400 without LOCKINPUT, and uncatalog the original database. For information on determining which utility to use, see Chapter 6, "RACF database utilities," on page 203.

      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 IRRUT200 with PARM=ACTIVATE, or IRRUT400 with LOCKINPUT to create a new backup. Uncatalog the original backup.

4. If you used IRRUT400, or IRRUT200 without PARM=ACTIVATE, issue RVARY ACTIVE for the new backup database.
5. If you used IRRUT400 with LOCKINPUT, 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 might 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 309.
- 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 233.

Chapter 2. Performance considerations

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 can minimize problems sometimes associated with hardware RESERVEs. An installation can explicitly convert hardware RESERVEs to global resource serialization ENQs.

Note: When enabled for sysplex data sharing and operating in data sharing or read-only mode, RACF always uses global resource serialization 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.

Selection of control unit and device

The choice of the DASD device and control unit for the RACF database can affect the performance and reliability of RACF and the system.

Guidelines: For best performance, follow these guidelines for selecting a DASD control unit and device for the RACF database:

- 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.
- Do not place the RACF database on the same control unit or device as other data sets that might have RESERVEs issued against them, such as catalogs and VSAM data sets.
- 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 87.

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 127 for more information.

RACF sysplex data sharing

RACF sysplex data sharing is an optional function that can:

- Facilitate system administration
- Provide consistent sysplex-wide security
- Improve performance in some environments

© Copyright IBM Corp. 1994, 2020

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 global resource serialization 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 91.

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 101 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 17.

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. If you specify PARM=ACTIVATE in your JCL, IRRUT200 activates the backup copy without allowing the RACF database to be updated between the copy and activate

operations, keeping the backup and primary data sets synchronized. 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 more information, see "RACF database split/merge/extend utility program (IRRUT400)" on page 233.

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 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, 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.

   Guideline: 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, 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 does not specify the same names or order, RACF ignores the names or order that is 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 (including 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, the default is 10 resident data blocks. If you have a data set name table, RACF uses the number that is 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 for the RACF database. RACF uses resident data blocks for performance reasons and to ensure that all processors reference the latest level of the blocks. When resident data blocks are used, some resource statistics might not be updated when there is high volume processing and the same profiles are updated by multiple operations. This is more common when the database is shared.

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 87. 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 74.

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 6, “RACF database utilities,” on page 203.

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 112.

Protecting data during its life cycle

Providing protection of data requires protecting data throughout its lifecycle. In addition to the access controls provided by RACF and DFSMS, IBM provides two mechanisms to provide additional levels of protection for DASD data sets:

- "Encryption of data at rest" on page 21
- "DFSMS considerations" on page 115
- "Erase-on-scratch" on page 22

These mechanisms are intended to address the threat of:

Table 1. Mechanisms intended to address threats to data during its life cycle

Threat	Addressed by Encryption of Data at Rest	Addressed by Erase on Scratch
The loss of physical control of the media, either through malevolent actions (such as theft) or non-malevolent actions (such as maintenance).	Yes	No
The movement of data within the storage device, such as for data recovery or performance	Yes	No
Residual data being read after deletion	No	Yes

Encryption of data at rest

Encryption of data at rest can be accomplished either through the use of encryption capable storage devices, such as the IBM DS8870 and the IBM TS3592, or through software such as the data set encryption facilities in DFSMSdfp or the IBM Encryption Facility.

Encryption capable devices implement in-line transparent encryption of data as it flows onto and off of the associated media. Encryption of data ensures that, in the event of physical loss of media (either through theft or replacement), the data is unreadable and its confidentiality is maintained. When data is moved within the boundaries of the device, it is encrypted throughout that process. Any residual data exposed on a drive is also encrypted. The ability to read data in unallocated space is eliminated because that residual data is encrypted. The IBM DS8870 implements Full Disk Encryption (FDE) whereby an external encryption key manager, such as the IBM Security Key Lifecycle Manager (IBM SKLM), creates and maintains the encryption keys on behalf of the DS8870.

Tape based datasets may be encrypted by the TS3592 devices. When directed to encrypt, by DFSMS for example, the drive contacts the IBM SKLM for an encryption key. The data is encrypted as it flows through the tape drive. The encryption keys are managed by the IBM SKLM.

DFSMSdfp provides the ability to specify an ICSF key-label in the RACF DFP segment. The key is associated with the dataset at the time of the datasets creation and is used to encrypt and decrypt data

when an application reads from or writes to the data set. For more information, see "DFSMS considerations" on page 115.

The IBM Encryption Facility provides a mechanism for encrypting datasets manually. Encryption keys maintained in RACF or ICSF are used by the Encryption Facility to encrypt from a source dataset to a target dataset. Access to the specific key material is required to decrypt the target dataset.

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 an attempt to read residual data. This protection means that no one can allocate a new data set at the same location, open it for input, and read your data.

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

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 modify a data set profile
  - Delete data sets using the RACROUTE REQUEST=AUTH postprocessing exit routines.

  Note:

  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.

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 7, "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 suggested 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 the following:

- 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.

If you are not sharing the database with a z/VM system, using SETROPTS RACLIST with a RACGLIST profile for the class provides the best performance with the lowest usage of common storage. For any

profiles that are shared with z/VM (for example, the VMMDISK or TERMINAL classes) you should evaluate the possible trade-offs between GENLIST and RACLIST, especially if the classes have a large number of discrete profiles. On z/VM, RACF has a limited amount of storage available for loading profiles, and it is all below 16 megabytes (24-bit addressing). Using GENLIST rather than RACLIST brings only the generic profiles into storage, rather than all the profiles. Therefore, if the class contains a large number of discrete profiles, using GENLIST significantly reduces the storage utilization. However, it can also hurt performance, especially for the z/OS system sharing the database.

RACLIST processing

The RACLIST operand on the SETROPTS command copies the base segments of generic and discrete profiles from the RACF database into storage. The profile copies are put in their own data space. Segments other than the base segments are not loaded into the 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
- APPCTP
- CRYPTOZ
- CSFKEYS
- CSFSERV
- DEVICES
- DIGTCRIT
- DIGTNMAP
- FIELD
- FSACCESS
- FSEXEC
- IDIDMAP
- NODES
- OPERCMDS
- PKISERV

- PROPCNTL
- PSFMPL
- PTKTDATA
- RACFHC
- RACFVARS
- RDATALIB
- SECLABEL
- SERVAUTH
- STARTED
- SYSMVIEW
- UNIXPRIV
- VTAMAPPL
- XCSFKEY

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

- ACCTNUM *
- ALCSAUTH
- APPCPORT
- APPCSI
- APPL *
- CBIND
- CDT *
- CONSOLE
- CPSMOBJ
- CPSMXMP
- DASDVOL
- DBNFORM
- DCEUUIDS
- DIGTCERT *
- DIGTRING
- DLFCLASS
- DSNR
- FACILITY *
- FCICSFCT
- INFOMAN
- JAVA
- JESINPUT
- JESJOBS
- JESSPOOL
- KEYSMSTR
- LDAPBIND *
- LFSCLASS
- LOGSTRM

- MGMTCLAS
- MQCMDS
- MQCONN
- NETCMDS
- PERFGRP *
- PKISERV
- PTKTVAL
- PRINTSRV *
- RRSFDATA *
- SDSF
- SERVER
- SMESSAGE
- SOMDOBJS
- STORCLAS
- SUBSYSNM
- SURROGAT
- TERMINAL *
- TMEADMIN
- TSOAUTH *
- TSOPROC *
- VMBATCH
- VMCMD
- VMDEV
- VMLAN
- VMNODE
- VMSEGMT
- WRITER

Important: For each class marked with an asterisk (*), you might incur performance degradation or missing function if you do not issue the SETROPTS RACLIST command when you define profiles in the class and activate it. For important details about each class, see *z/OS Security Server RACF Security Administrator's Guide* (for classes used for RACF functions) or the appropriate program documentation.

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 BASE, SESSION, and ICSF segments of 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.

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 that is named on the command plus any other classes that share the same POSIT value. See *z/OS Security Server RACF Security Administrator's Guide* for more information.

Because SETROPTS RACLIST loads only the BASE, SESSION, and ICSF segments into the data space, if you delete a profile, you should issue a SETROPTS RACLIST(*classname*) REFRESH command immediately. If you do not, the copy of the segments in the data space and the segments in the database do not match, and might cause unexpected results. For example, if you delete a profile, all of its segments are deleted from the RACF database, but until you issue a SETROPTS RACLIST(*classname*) REFRESH command, the copy of the original segments remains in the data space. From RACF's point of view, the profile still exists, because the segments are still in the data space, but if RACF tries to reference a non-base segment for the profile, it no longer exists in the database.

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

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

If only BASE, SESSION, and ICSF segment information is updated, wait to issue the SETROPTS RACLIST(*classname*) REFRESH command until you want the changes to take effect. If these segments are updated with other segments, issue a SETROPTS RACLIST(*classname*) REFRESH command immediately.

For some classes, selected profile data is kept in storage. Changes to these profiles might not be active until a refresh is done. A SETROPTS RACLIST(*classname*) REFRESH ensures that profile data is consistent. An example of this type of class is PTKTDATA.

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
- CPSMOBJ
- DASDVOL
- DCEUUIDS
- DSNR
- FACILITY
- FIELD
- GXFACILI
- ILMADMIN
- INFOMAN
- JESJOBS
- KEYSMSTR
- LOGSTRM
- PRINTSRV
- RACFEVNT
- RRSFDATA
- SDSF
- TERMINAL
- TMEADMIN
- VMBATCH
- VMCMD
- VMDEV
- VMLAN
- VMMDISK
- VMNODE
- VMRDR
- VMSEGMT
- XFACILIT

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 guidelines:

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

  Guideline: 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.

  Guideline: 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 their password or password phrase, and each time the user enters the correct password or password phrase after having previously entered an incorrect one. If SETROPTS NOINACTIVE is in effect, the update for the first logon of the day is not recorded in the backup database.

  At all other times these statistics are recorded only to the primary database. When resident data blocks are used, some resource statistics might not be updated when there is high volume processing and the

same profiles are updated by multiple operations. This is more common when the RACF database is shared.

Guideline: Although INITSTATS affects performance because of I/O to the database, keep INITSTATS on. You can then use the SETROPTS operands INACTIVE and PASSWORD. For more 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.

Installations with many RACF users who log on to various applications many times throughout the day, may choose to limit the impact of INITSTATS statistics recording for certain applications. These installations can specify that statistics for certain applications should be recorded for each user only once a day. If daily user statistics are set for a particular application, then, when the application issues a RACROUTE REQUEST=VERIFY, statistics are recorded only if no recording has been performed for the particular user that day. This can lessen the impact of serialization on the RACF database. To specify daily user statistics for an application, that application must specify the APPL operand on the RACROUTE REQUEST=VERIFY macro. When an application specifies the APPL operand, the system administrator can use a profile in the APPL class to control which users can access the application, and can limit INITSTATS processing to collect daily statistics only. To limit INITSTATS processing to collect daily statistics only, the system administrator specifies the string 'RACF-INITSTATS(DAILY)' in the APPLDATA field of the APPL class profile. For more information, refer to *z/OS Security Server RACF Security Administrator's Guide*.

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 password phrase, or the user ID is revoked.

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

Callers of RACROUTE REQUEST=VERIFY may also realize improved performance when the impact of INITSTATS statistics recording is lessened. An installation can specify that, when a user is verified by a particular application, statistics should be recorded only if no recording has been done for that user that day. When an application specifies the APPL operand on the RACROUTE REQUEST=VERIFY request, the system administrator can use a profile in the APPL class to control which users can access the application, and can limit INITSTATS processing to collect daily statistics only. To limit INITSTATS processing to collect daily statistics only, the system administrator specifies the string 'RACF-INITSTATS(DAILY)' in the APPLDATA field of the APPL class profile. For more information, refer to *z/OS Security Server RACF Security Administrator's Guide*.

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 74.

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 73. 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.

Program signing and signature verification

RACF supports signing and verification of program objects. This means that IBM or a vendor can ship program objects containing a digital signature (including the digital certificate chain for the user who performed the program bind). An installation can then choose to verify the integrity of these program objects (for example, the System SSL modules) when they are loaded into virtual storage.

Installations that choose to use signature verification experience some performance overhead when the signature of a signed program object is verified before being loaded into memory. However, most of this overhead is from validating the certificate chain that was used during the signing process and can be lessened using the virtual lookaside facility (VLF) to cache certificates after they are validated. Most of the overhead from validating a certificate chain is then incurred only once, the first time a program object from a particular signer is loaded. Subsequent calls to load any program object from the same signer is able to avoid the performance overhead.

For information about enabling VLF caching, see “VLF considerations for program signature verification” on page 75. For information about initializing program signature verification, see “Initializing RACF verification of signed programs (IRRVERLD)” on page 125. For implementation details, see Enabling RACF to verify signed programs in *z/OS Security Server RACF Security Administrator's Guide*.

Using generic profiles

In each address space, RACF keeps 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 class descriptor table (CDT) entry for that class (assuming the class is not RACLISTed in some way). By default, RACF keeps four lists, but you can use the RACF SET command with the GENERICANCHOR operand to specify that RACF keep a larger number of lists.

The KEYQUAL value in the class descriptor table (CDT) entry for a class specifies the number of matching qualifiers that RACF uses when loading generic profile names to satisfy an authorization request if a discrete profile does not exist for the resource. For example, if the value of KEYQUAL is 2, all generic profile names whose two highest-level qualifiers match the two highest-level qualifiers of the entity name are loaded into the user's storage when the user requests access to a resource. If the value of KEYQUAL is 0, profile names for the entire class are loaded and searched. You specify the number of key qualifiers for a class when you define the class. For more information about key qualifiers, see the description of the KEYQUAL operand on the ICHERCDE macro in *z/OS Security Server RACF Macros and Interfaces* or the description of the KEYQUALIFIERS operand on the RDEFINE command in *z/OS Security Server RACF Command Language Reference*.

RACF keeps the lists in 64-bit memory objects. The GENERICANCHOR settings for the system or a job and the number of generic profiles in the referenced non-RACLISTed classes or data set high-level qualifiers

determine the amount of storage RACF uses. The user's or job's MEMLIMIT does not restrict the amount of storage that RACF uses for the lists.

When RACF needs to reference a set of generic profiles that are not present in the address space, and the maximum number of generic profile lists have been loaded for the address space, RACF deletes the oldest list and replaces it with the new list. 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. Increasing the number of lists can improve performance when an application references more than four different data set high-level qualifiers or general resource classes. However, increasing the number of lists can increase the amount of virtual storage used, especially if you have many profiles for one or more of the resource classes or high-level qualifiers.

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.

Customizing the number of lists of generic profiles that RACF maintains

Use the GENERICANCHOR operand on the SET command to customize the number of lists of generic profiles that RACF maintains. You can set a system-wide number, and numbers for specific jobs. You must have the RACF subsystem active in order to use the RACF SET command, and you should configure it to process a RACF subsystem parameter library member automatically during system IPL, and place your SET GENERICANCHOR commands in that member. RACF does not remember the SET options across IPLs. You must specify them during or after IPL on any system where you want them to apply, and using the RACF subsystem parameter library ensures that this happens. For information about activating the RACF subsystem, see “Activating the RACF subsystem” on page 77. For information about using the RACF parameter library, see “The RACF parameter library” on page 184. For information about the SET command, see *z/OS Security Server RACF Command Language Reference*.

Example: You want to change the number of lists maintained to eight system-wide. Add the following command to the RACF parameter library:

```
SET GENERICANCHOR(SYSTEM COUNT(8))
```

or enter the following operator command, where # is the command character defined for the RACF subsystem:

```
#SET GENERICANCHOR(SYSTEM COUNT(8))
```

Example: You want to change the number of lists maintained for a job JOBXYZ to 10. Add the following command to the RACF parameter library:

```
SET GENERICANCHOR(JOBNAME(JOBXYZ) COUNT(10))
```

or enter the following operator command, where # is the command character defined for the RACF subsystem:

```
#SET GENERICANCHOR(JOBNAME(JOBXYZ) COUNT(10))
```

Example: You want to change the number of lists maintained to 10 for JOBXYZ, to five for JOB PDQ, and to five for job WALT. Create a RACF parameter library member named, for example, IRROPT22 containing the following commands:

```
SET GENERICANCHOR(JOBNAME(JOBXYZ) COUNT(10))
SET GENERICANCHOR(JOBNAME(JOBPDQ) COUNT(5))
SET GENERICANCHOR(JOBNAME(WALT) COUNT(5))
```

Then, in RACF parameter library member IRROPT00, add the command:

```
SET INCLUDE(22)
```

The commands in IRROPT00 are run each time the RACF subsystem starts (unless you have specified a different parameter library member in your RACF procedure). To make the new commands in IRROPT22 run immediately, enter the following command as an operator command, where # is the command character defined for the RACF subsystem:

```
#SET INCLUDE(22)
```

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.

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 an alias index if at application identity mapping stage 3, or by accessing one profile in the UNIXMAP class. RACF adds the mapping to VLF if it finds it in the UNIXMAP class or alias index.

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 74. 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 75.

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 35.

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

RACF database configuration

Overview

RACF database configuration consists of the following three parts:

The data set name table
Describes the data sets that comprise the RACF database.

The database range table
Determines in which data set of the RACF database RACF places each profile.

Sysplex communication
Describes sysplex configuration options.

The data set name table

The data set name table describes the data sets that comprise the RACF database. The table contains one or more pairs of primary and backup data set names.

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

Primary data set name

The required data set name in which RACF profiles are stored. If your installation has specified this field with an asterisk (*), RACF prompts the operator to supply the data set name during RACF initialization.

The data set name must be the real name of the data set. Do not specify an alias.

Backup data set name

The data set is an optional backup of the primary data set. If your installation specifies this field with an asterisk (*), RACF prompts the operator. A blank data set name field indicates the absence of a backup data set for the associated primary data set for this IPL.

The data set name must be the real name of the data set. Do not specify an alias.

Backup options

The following are update options that you can specify for the backup data set:

- No updates are duplicated on the backup data set. This is the default setting.
- All updates, but no statistics are to be duplicated on the backup data set. This is the recommend setting if you provide a data set name table.

  If SETROPTS INITSTATS is active, a limited subset of statistics is maintained on the backup:

  – The first time of each day that the user logs on when SETROPTS INACTIVE is in effect.
  – The time and date of when a password or password phrase changes during log on.
  – The time and date a user enters a correct password or password phrase after having entered an incorrect one.

  SETROPTS INITSTATS enables RACF to continue processing revoke dates if you need to switch to your backup database. If you have SETROPTS NOINACTIVE in effect, you are not revoking users for inactivity. Therefore, the statistics for the first logon of the day are not recorded in the backup database..
- All updates, including statistics, are duplicated on the backup data set.

© Copyright IBM Corp. 1994, 2020

Resident data-block count used for buffers

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.

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, 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, including index blocks) resident. An installation can specify from 0 to 255 resident data blocks. 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 during 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 × .2 × 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.

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 does not 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 does not 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.

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 102 for information on coupling facility structure definitions.

The database range table

The range table determines in which data set of the RACF database RACF places each profile. The range table can contain one or more ranges. Each range in the range table points to an entry in the data set name table.

The range table consists of a list of starting points for each range. The end of a range is determined by the start value of the next range. The first range start value must contain all binary zeros and all subsequent ranges must be in ascending order.

In the data set name table, RACF assigns a number to each primary and backup data set pair that is defined. Each range must reference an entry number in the data set name table so that RACF knows which data set to read or write a profile to.

RACF provides a default range table that contains one range of hexadecimal zeros for the start value that points to the first entry in the data set name table. This table assumes that the RACF database has one data set containing all profiles.

When you are constructing a range table, you must also consider how RACF generates the internal form of each profile. The RACF manager uses only the internal forms of these profile names; therefore, you must 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 hexadecimal values in effect for a particular alias field:

Table 2. 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
SERVAUTH profile with an IP lookup value	X'050C02' followed by lookup value

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*.

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 sharing a common RACF parmlib member or common ICHRDSNT load module.
- The number of resident data blocks specified for buffers can differ between systems in the data sharing group.
- 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 might be overridden at IPL time to the mode in use by the other members of the data sharing group.

For more information about sysplex communications and sysplex data sharing options, see "Sysplex considerations" on page 91.

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.

Specifying RACF database options

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

- The IRRPRMxx PARMLIB member is specified via the "RACF=xx" system initialization option. In this member, or set of members, you can define the database sharing/sysplex communications mode, data set name table, resident data blocks (buffers), and database range table.
- If you are not using an IRRPRMxx PARMLIB member, you can define the data set name table (in addition to database sharing/sysplex communications mode and resident data blocks) in a load module named ICHRDSNT. You can also define the database range table in a load module named ICHRRNG.

IRRPRMxx parmlib member

The data set name table and the database range table can be specified by using the parmlib member, IRRPRMxx.

The parmlib member, IRRPRMxx, contains data for the RACF data set name table, range table, and sysplex communications options.

If you want to specify more than one parmlib members, see "Multiple parmlib members" on page 43.

The ICHRRNG exit is required on the system even when using the IRRPRMxx parmlib member. This is required due to the order of processing at IPL-time and how early the ICHRRNG module must be loaded.

IRRPRMxx parmlib parameter syntax

The IRRPRMxx parmlib parameter syntax is as follows:

```
DATABASE_OPTIONS
     SYSPLEX(NOCOMMUNICATIONS | COMMUNICATIONS | DATASHARING)
     DATASETNAMETABLE
       ENTRY
         PRIMARYDSN(data-set-name)
         BACKUPDSN(data-set-name)
         UPDATEBACKUP(ALL | NONE | NOSTATS)
         BUFFERS(value)
     RANGETABLE
       START(start-value [CHAR | HEX]) ENTRYNUMBER(entry-sequence-number)
```

The IRRPRMxx parmlib parameter descriptions are as follows:

DATABASE_OPTIONS
Header statement for the RACF initialization portion of the parmlib member. This parameter is required for each parmlib member that is processed.

SYSPLEX
Specifies the sysplex environment that you want.

NOCOMMUNICATIONS | NOCOMM
Specifies that RACF is not enabled for sysplex communications.

COMMUNICATIONS | COMM
Specifies that RACF is enabled for sysplex communications and requests non-data sharing mode at IPL.

DATASHARING
Specifies that RACF is enabled for sysplex communications and requests data sharing mode at IPL.

This parameter is optional. If not specified, `SYSPLEX(NOCOMMUNICATIONS)` is the default value.

DATASETNAMETABLE | DSNT
Header statement that marks the beginning of the data set name table. This table can contain multiple entries. The maximum is 90 primary and backup data set pairs.

This parameter is required when defining a data set name table. It is optional if the default data set name table is sufficient. The default data set name table includes a single primary and backup data set pair, for which the operator is prompted. This dataset pair has a default of 255 resident data blocks for buffers and no data duplication to the backup data set.

ENTRY
Header statement that marks the beginning of a single primary and backup data set pair in the data set name table. There can be multiple ENTRY statements to mark each primary and backup data set pair.

This parameter is required for each entry in the data set name table.

PRIMARYDSN(*data-set-name*) | PRIMARY(*data-set-name*) | PRIM(*data-set-name*)
Specifies the name of a primary data set. The data set name must follow data set naming conventions. It can either be a quoted string or an unquoted string. If the data set name is an asterisk, the operator is prompted for a data set name at initialization time.

This parameter is optional. If not specified, the operator is prompted for a primary data set name.

BACKUPDSN(*data-set-name*) | BACKUP(*data-set-name*) | BACK(*data-set-name*)
Specifies the name of a backup data set. The data set name must follow data set naming conventions. It can either be a quoted string or an unquoted string. If the data set name is an asterisk, the operator is prompted for a data set name at initialization time. If the data set name is a blank, the backup is not defined and the operator is not prompted.

This parameter is optional. If not specified, the operator is prompted for a backup data set name.

UPDATEBACKUP | UPDATEBACK
Specifies the update option that you want for the backup data set.

ALL
All updates, including statistics, are to be duplicated on the backup data set.

NONE
No updates are to be duplicated on the backup data set.

NOSTATS
All updates, but no statistics, are to be duplicated on the backup data set.

This parameter is optional. If not specified, `UPDATEBACKUP(NONE)` is the default value.

BUFFERS(*value*) | BUF(*value*)
Specifies the number of data blocks to be kept resident for the primary data set. If the system is enabled for sysplex communications, 20% of this value is used for the backup data set. The valid values are 0 - 255.

This parameter is optional. If not specified, the default value is 255.

RANGETABLE | RANGE | RNG
Header statement that marks the beginning of the range table. This table can contain multiple entries. The first start entry must be binary zeros and must be arranged in ascending order. The first entry can appear as follows:

```
START('00' HEX) ENTRYNUMBER(1)
```

If only one entry exists in the data set name table, no range table is required.

When more than one range exists in the range table, each range must point to an entry in the data set name table. When more than one entry exists in the data set name table, each entry must be referenced by at least one range in the range table; otherwise, an error occurs.

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. For more information about internal names, see "Internal profile naming for general resource classes" on page 39.

The RANGETABLE parameter is optional. If not specified, a default range table is defined containing a start value of binary zeros that points to the first entry in the data set name table.

START(*start-value* [CHAR | HEX])
Specifies the starting range for a single primary and backup data set pair. The start value can be a string of hexadecimal or alphanumeric characters with or without single quotation marks. Single quotation marks are required for the following:

- Lowercase characters
- Any blanks included in the string
- A START value that spans multiple lines
- A START value that contains a single quotation mark within the string. The single quotation mark within the string must immediately be followed by another single quotation mark.

If lowercase characters are not enclosed in single quotation marks, they are folded to uppercase characters.

The START parameter is optional. If not specified, the default is a single entry of binary zeros pointing to the first entry in the data set name table.

CHAR
Specifies the start value that is interpreted as character data. The maximum length of a *start-value* is 44.

HEX
Specifies the start value that is interpreted as hexadecimal data. The accepted characters are 0 - 9 and uppercase characters A - F. The maximum length of a *start-value* is 88 hexadecimal characters. An even number of characters must be specified.

If not specified, the default is CHAR.

ENTRYNUMBER(*entry-sequence-number*) | ENTRYNUM(*entry-sequence-number*)
Specifies the sequence number (the data set's relative position in the data set name table) of the entry containing the primary and backup data set pair that is associated with the range start value. The valid values are 1 - 90.

The START and ENTRYNUMBER parameters must be specified in pairs.

Syntax rules for IRRPRMxx

The following are syntax rules for the parmlib member, IRRPRMxx:

- Columns 1 - 71 are used for data; Columns 72 - 80 are ignored.
- Comments can appear in columns 1 - 71 and must begin with /* and end with */.
- A section (data set name table, range table or sysplex consideration options) cannot exist multiple times within the same member.
- One or more statements can be entered on a single line or one statement can span multiple lines. For instance, a parameter can appear on one line and its associated value can appear on another. However, the keyword parameter itself cannot span lines. It is recommended that one keyword per line is specified for debugging purposes.
- The continuation of a parameter value is allowed if the parameter value is enclosed in quotation marks (such as the value of the START parameter). There is no explicit continuation character. The quoted string consists of all data up to column 71 and continues in column 1 on the next line.
- At least one blank is required between each parameter, however, not between the parameter and its value. The right parenthesis after the specified parameter value is a sufficient delimiter.
- The parameters and parameter values can be entered in uppercase, lowercase, or in single quotation marks. Any values that are enclosed in single quotation marks are processed as entered. Otherwise, the system converts all characters to uppercase.
- A maximum of 3 parmlib members can be specified on the RACF= parameter during initialization.

Multiple parmlib members

When specifying multiple parmlib members, the data within the data set name table and the range table cannot be split between members. The data set name table and the range table can exist in individual members. However, having some primary and backup data set pairs in one member, and the remaining data set pairs in a different member is not permitted. The same rules apply to the ranges within the range table. The SYSPLEX options can be specified in a separate member.

A maximum of 3 parmlib members can be specified in the following format:

```
RACF=(aa,bb,cc)
```

If a section (data set name table, range table, or sysplex consideration options) is specified in more than one parmlib member, only the first occurrence is processed. All duplicates are ignored. If a section is specified more than once in a single member, an error message is issued.

If parmlib members are specified, but no data is found in any of the members, an error message is issued. If multiple members are specified and one or more members are empty, the empty members are ignored.

Error processing

If any errors occur during the processing of a parmlib member, an error message is issued to indicate the problem and where it occurred. Parsing continues and additional error messages can be issued as identified.

If errors are detected at the time of IPL, no data from any parmlib member is used. In addition the load modules, ICHRDSNT and ICHRRNG, are not used. The operator is then prompted to specify the name of a single primary and backup data set, or NONE if no primary and backup is to be used. The parmlib members specified at IPL time are documented in the system log along with the data set name where it is found.

Sharing a database

All systems that are sharing a database must use the same or a compatible data set name table and range table. One way to ensure that each Version 2 Release 3 or higher uses the same tables, is to put the RACF parmlib member in a common parmlib data set.

It is possible to use a combination of parmlib members and load modules (ICHRDSNT and ICHRRNG), if the table definitions are compatible.

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

IRRPRMXX parmlib sample

A RACF parmlib member sample is provided in SYS1.SAMPLIB(IRRPRMXX).

ICHRDSNT load module

An alternative to a RACF parmlib member, ICHRDSNT is an installation-defined load module that can be used to define the data set name table. This table contains entries that describe 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.

All systems that are sharing a database must use the same or a compatible data set name 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 data set name table matches the data set name table that is used by the rest of the data sharing group. If there is a mismatch, it uses the table that is used by the group.

ICHRDSNT 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 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.

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. 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.

....1
Formerly controlled the resident data-block option for the primary database. 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, *z/OS Program Directory* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

ICHRDSNT 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.

Note:

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

ICHRRNG load module

An alternative to a RACF parmlib member, ICHRRNG is an installation-defined load module that can be used to define the database range table. 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 that are sharing a database must use the same or a compatible database range table. One way to ensure that each system uses the same table is to put the table in a common LPA library.

If RACF is enabled for sysplex communication, it verifies at IPL that a local system's range table matches the range table that is used by the rest of the data sharing group. If there is a mismatch, it uses the table that is 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 that the RACF database has one data set containing all profiles. If you want 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.

ICHRRNG 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. For more information about internal names, see "Internal profile naming for general resource classes" on page 39.

ICHRRNG 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 4. ICHRRNG example for three data sets

Other considerations

Effects of not using a data set name table

As described, a data set name table can be defined by either a RACF parmlib member or ICHRDSNT load module. If at IPL RACF does not find a data set name table and the master JCL does not point to a RACF database with a SYSRACF DD statement, RACF prompts the operator for primary and backup data set names. However, not using a data set name table results in a primary database with only 10 data blocks, an inactive backup database, and poor system performance.

Guideline: Use a data set name table instead of MSTJCL*xx*, because MSTJCL*xx* recognizes only the primary database when the SYSRACF DD statement is provided.

Emergency data set name tables

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

You might, for example, have your normal data set name table that is defined in the parmlib member IRRPRMOK and is used when RACF=OK is specified at system initialization. Create a second parmlib member, IRRPRMEM, with an identical data set name table. Replace the primary and backup data set names with an asterisk (*). When this parmlib member is indicated by RACF=EM at system initialization, the operator is prompted to enter the names for the emergency primary and backup data sets.

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

A similar procedure can be done with an ICHRDSNT, by making a copy of the module and replacing the data set names with an asterisk (*). When you are IPLing with the emergency ICHRDSNT load module, reference a link list concatenation that includes the emergency ICHRDSNT load module.

Sysplex considerations

If you use the above procedure when your current RACF system is enabled for sysplex communication, and the emergency data set name table 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 do not 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.

It is recommended to have an emergency data set name table that has the sysplex communication and data sharing option OFF. You can now IPL a single system in non-sysplex-communication/non-data sharing mode, which might be necessary in some recovery scenarios.

Attention: This data set name table 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/non-data sharing environment is incompatible with serialization in a sysplex-communication/data sharing environment. Bringing up a non sysplex-communication/non-data sharing system against your main sysplex database is likely to result in database corruption.

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.

The class descriptor table has two parts:

- The static class descriptor table.

  If you make a change to the static class descriptor table, you must re-IPL to have the change take effect on your system. This part of the class descriptor table contains two load modules:

  – ICHRRCDX contains the class entries supplied by IBM. 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 *z/OS Security Server RACF Security Administrator's Guide*.

  – ICHRRCDE is an optional module that contains installation-defined class entries. You define classes in ICHRRCDE using the ICHERCDE macro. For information, see ICHERCDE macro in *z/OS Security Server RACF Macros and Interfaces*.

- The dynamic class descriptor table.

  This optional portion of the class descriptor table contains installation-defined class entries built from the CDT general resource class. You can define classes in the dynamic class descriptor table using RDEFINE and RALTER commands. If you make a change to the dynamic class descriptor table, you do *not* need to re-IPL to have the change take effect. Instead, issue the command SETROPTS `RACLIST(CDT) REFRESH`. For information about the dynamic class descriptor table, see *z/OS Security Server RACF Security Administrator's Guide*.

  Restriction: If you have applications or vendor products that use dynamic classes, they must use the RACROUTE REQUEST=STAT interface to process information for dynamic classes (for example, to check if a class is active). If your application or vendor product uses the RACSTAT macro or the RCVTCDTP pointer in the RCVT control block to locate a dynamic class, it will not work properly.

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 can 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.

Beginning with z/OS V1R8, you can define a class with an attribute that disallows generic profile processing. For information about using a class that does not allow generic profile processing when the database is shared by systems that do and do not support this attribute, see “Considerations for classes that do not allow generic profile processing” on page 9.

If you have systems at different releases of z/OS, you can share a database between them 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 a z/OS V1R4 system and a z/OS V1R6 system, you do not have to add the new RACF classes to the class descriptor table on the z/OS V1R4 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.

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. There are 1024 POSIT numbers, of which numbers 19–56 and 128–527 are available for your installation's use. Numbers 0–18, 57–127, and 528–1023 are reserved for IBM's use. Classes requiring better performance should be placed towards the beginning of the table.

Adding installation-defined classes to the static class descriptor table

There are two ways for an installation to define resource classes:

- Define them in the dynamic class descriptor table, using RDEFINE and RALTER commands. For information on how to do this, see *z/OS Security Server RACF Security Administrator's Guide*.
- Define them in the static class descriptor table, using the ICHERCDE macro, as described in the remainder of this section.

Guideline: Define your classes in the dynamic class descriptor table, to avoid the need to re-IPL.

An installation can add, modify, or delete installation-defined entries in the static class descriptor table using the ICHERCDE macro. 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.

The ICHERCDE macro generates class entries for the RACF static class descriptor table. Each entry in the installation-defined static 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 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.
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*, available at CICS Transaction Server for z/OS (www.ibm.com/support/knowledgecenter/SSGMGV/welcome.html).

Changing an installation-defined class in the static 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 will not receive 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 are 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.
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 static class descriptor table

You can delete a class entry from the static 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 all systems that used the class.

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, and RACLIST. 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 is optional, and if present is the module ICHRFR01. The entries in ICHRFR01 can be for installation-defined resource classes and combinations of requestor and subsystem. The RACF router assumes that if there is no entry in the RACF router table for a combination of resource class, requestor, and subsystem, that combination is to be treated as if ACTION=RACF was specified in the router table, and RACF is called on each invocation of the RACROUTE macro.

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 the system for the change to take effect.

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

ENF signals

RACF can send an ENF signal to listeners in the following situations:

- When a SETROPTS RACLIST command affects in-storage profiles used for authorization checking, RACF sends a type 62 ENF signal to listeners.
- When a CONNECT, REMOVE, ALTUSER REVOKE, DELUSER, or DELGROUP command has affected a user's group connections, RACF sends a type 71 ENF signal to listeners.
- When a PERMIT, RDEFINE, RALTER, or RDELETE command has affected a user's or group's authorizations to resources, RACF sends a type 79 ENF signal to listeners. This signal is issued only for classes that have been defined in the RACF Class Descriptor Table with the SIGNAL=YES option.

Listeners of these signals should follow the guidelines documented in *z/OS MVS Programming: Authorized Assembler Services Guide* on coding listener exit routines, particularly:

- Avoid such time-consuming processing as obtaining large amounts of storage through the GETMAIN macro, issuing WAITs or issuing SVCs that issue the WAIT macro, and performing I/O operations.
- Avoid requests for the local lock.
- Avoid using multiple listener user exits.

Type 62 ENF signals

RACF can send an ENF signal to listeners when a SETROPTS RACLIST command affects in-storage profiles used for authorization checking. RACF sends a signal when a SETROPTS RACLIST, SETROPTS NORACLIST, or SETROPTS RACLIST REFRESH command is issued for a class, activating, deactivating, or updating the profiles. Signals are sent for a class in the static class descriptor table if SIGNAL=YES was specified on the ICHERCDE macro that defined the class. Signals are sent for a class in the dynamic class descriptor table if SIGNAL(YES) was specified on the CDTINFO keyword of the RDEFINE or RALTER command that defined the class.

Guideline: Only specify SIGNAL=YES when the documentation for a product or application states that you should do so for its user-defined class. Turning on signals for classes that have no listeners, depending on how many classes you do this for, and how many SETROPTS RACLIST, SETOPTS RACLIST REFRESH, and SETOPTS NORACLIST commands are issued for those classes, can result in unnecessary processing by all exits that listen for ENF 62 signals.

When the in-storage profiles for such a class are activated, deactivated, or updated, RACF sends a type 62 ENF signal to listeners, with a parameter list mapped by IRRPENFP in SYS1.MACLIB. Qualifier byte 1 indicates a SETROPTS RACLIST, qualifier byte 2 indicates a SETROPTS RACLIST REFRESH, and qualifier byte 3 indicates a SETROPTS NORACLIST. The parameter list contains the class name.

RACROUTE REQUEST=LIST,GLOBAL=YES does not cause an ENF signal to be issued. When classes that are GLOBAL=YES ONLY RACLISTed are refreshed with SETROPTS RACLIST REFRESH, RACF issues an ENF signal. If they are SETROPTS NORACLISTed, RACF issues the ENF signal only on a system that has the class GLOBAL=YES RACLISTed. Avoid using SETROPTS NORACLIST in the case of a RACROUTE REQUEST=LIST,GLOBAL=YES class unless everyone has disconnected from the dataspace. At that point, it is unlikely that anyone is listening for an ENF signal.

RACF sends no signals when an application issues RACROUTE REQUEST=LIST,GLOBAL=NO.

For a class in the static class descriptor table, if RACLIST=DISALLOWED is specified for a class, no signal is sent even if SIGNAL=YES is specified. For a dynamic class, you should not specify RACLIST(DISALLOWED) with SIGNAL(YES); if you do, the class is not added to the dynamic class descriptor table.

ENFREQ listener routines for the Type 62 ENF signal, should listen as XSYS=NO.

Table 3. ENF 62 event code			
Description	**Qualifier**	**Parameter list passed to user exit**	**Exit type/ Cross-system capable**
A RACF SETROPTS RACLIST command has affected in-storage profiles used for authorization requests in a class designated as SIGNAL=YES or SIGNAL(YES) in the RACF class descriptor table. The class affected is in the parameter list in field IRR_ENFCLASS.	The qualifier (QUAL) has the following format: • BYTE1 X'80' SETROPTS RACLIST has taken place • BYTE2 X'80' SETROPTS RACLIST REFRESH has taken place • BYTE3 X'80' SETROPTS NORACLIST has taken place	Mapped by IRRPENFP in SYS1.MACLIB. (See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).)	EXIT or SRBEXIT/ NO

Type 71 ENF signals

RACF sends a type 71 ENF signal to listeners when certain user or group changes occur. Such changes may affect resource authorization.

The following table will demonstrate the exact events.

Table 4. ENF 71 event code			
Description	**Qualifier**	**Parameter list passed to user exit**	**Exit type/ Cross-system capable**
RACF sends a type 71 ENF signal to listeners when certain user or group changes occur. Such changes may affect resource authorization. The user that is affected is in the parameter list in field IRR_ENF2USER. The group affected is in the parameter list in field IRR_ENF2GROUP. Control flags that are used to provide greater granularity for the listeners are in the parameter list in field IRR_ENF2Flags.	The qualifier (QUAL) has the following format: • BYTE1 X'80' CONNECT command • BYTE1 X'40' REMOVE command • BYTE1 X'20' ALTUSER REVOKE command • BYTE1 X'10' DELUSER command • BYTE1 X'08' DELGROUP command • BYTE1 X'04' RACROUTE REQUEST=VERIFY command • BYTE2 - 4 reserved	Mapped by IRRPENF2 (See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).)	EXIT / YES

Type 79 ENF signals

RACF sends a type 79 ENF signal to listeners when a PERMIT, RDEFINE, RALTER, or RDELETE command has affected a user's or group's authorizations to resources. When a user's or group's authorizations to resources has been affected by a PERMIT, RDEFINE, RALTER, or RDELETE command, this may affect the resource authorization. This signal is issued only for classes that have been defined in the RACF class descriptor table when the SIGNAL=YES is specified.

Table 5. ENF 79 event code			
Description	**Qualifier**	**Parameter list passed to user exit**	**Exit type/ Cross-system capable**
A RACF PERMIT, RDEFINE, RALTER, or RDELETE command has affected a user's or group's authorizations to resources. The user that is affected is in the parameter list in field IRR_ENF3_UserID. The class in which the modified profile belongs is in the parameter list in field IRR_ENF3_ClassName. The length of the affected profile name is in the parameter list in field IRR_ENF3_ProfName_Length. The name of the affected profile is in the parameter list in field IRR_ENF3_ProfName. Control flags that are used to provide greater granularity for the listeners are in the parameter list in field IRR_ENF3_Flags.	The qualifier (IRR_ENF3_QualCode) has the following format: • BYTE1 X'80' PERMIT command • BYTE1 X'40' RDEFINE command • BYTE1 X'20' RALTER command • BYTE1 X'10' RDELETE command • BYTE2 - 4 reserved	Mapped by IRRPENF3 (See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).)	EXIT / YES

Password authentication options

RACF provides three algorithms for authenticating passwords and password phrases: the masking algorithm, the Data Encryption Standard (DES) algorithm, and Key Derivation Function with Advanced Encryption Standard (KDFAES). The masking algorithm is the original algorithm provided with RACF. The RACF DES algorithm provides a higher level of security than the masking algorithm. The KDFAES algorithm provides the highest level of security, and is designed to be resistant to offline attacks. The DES algorithm is the default algorithm when you install RACF on your system.

Guideline: Use the KDFAES algorithm.

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 encryption algorithms (KDFAES and DES)

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.

A RACF 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 encryption algorithms work

When a user changes a password or password phrase, RACF treats the new value as an encryption key to transform the RACF user ID into an encoded form that it stores on the database. The password or password phrase is not stored.

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

Notes:

1. If two or more systems share the RACF database, they must all use the same password 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.
2. 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.
3. 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.

Guideline: A network is only as secure as its weakest point of entry. Use the KDFAES authentication algorithm on all systems in an RRSF network, to reduce the risk of compromising a password that can be used on multiple systems.

The RACF KDFAES algorithm

The KDFAES algorithm is used to encrypt passwords and password phrases, but not OIDCARD data. It is designed to be resistant to offline attacks by incorporating the following properties:

- Each instance of a RACF password injects randomly generated text into the encryption process. This prevents the use of pre-computed password hashes. That is, an offline attack must perform the full encryption process for every password guess, as opposed to simply comparing the password hash against a list of pre-computed values. This slows down the attack, making it take much longer to guess passwords.
- Thousands of hash operations are performed against the password and random text in order to generate a key which is then used to encrypt the user ID. This also serves to slow down an offline attack, which must perform the same number of operations for each password guess. However, the authorized user logging on to the system using his clear text password will not notice the increased overhead.

KDFAES is enabled using the SETROPTS command, not the ICHDEX01 exit: `SETROPTS PASSWORD(ALGORITHM(KDFAES))`

Note: Review the "Planning Considerations for enabling KDFAES" on page 59 prior to enabling KDFAES.

When KDFAES is enabled, existing DES passwords will continue to evaluated properly by RACF. User passwords do not need to be changed. When the user next changes his password, it will be encrypted using the KDFAES algorithm. The PWCONVERT operand of the ALTUSER command can be used to transform a DES password (but not a password phrase) into a KDFAES password without requiring the password to be changed.

Keep in mind that if you have backups of the RACF database containing passwords that were encrypted using DES or masking, they are more susceptible to offline attacks. If the hash represents the same clear text password as the user's current password, and an attacker is able to guess the value, it can be used to log on to the user's account even if the current password is encrypted using KDFAES. The EXPIRED operand of the ALTUSER command can be used to mark a password as expired, requiring it to be changed at the next logon. This can help accelerate the password change process.

Note: If you have passwords that were encoded using the masking algorithm, these will need to be changed. They will not be properly evaluated when KDFAES is enabled, and cannot be converted to KDFAES using the PWCONVERT function.

The RACF DES algorithm

The DES algorithm, unlike KDFAES, is used to encrypt OIDCARD data, in addition to passwords and password phrases.

The two-step method of password authentication

RACF provides a two-step method of authentication for passwords, password phrases, and OIDCARD data, originally intended to allow installations to migrate from the masking algorithm to the DES algorithm. To have RACF use the two-step method, write an ICHDEX01 exit (in the link pack area) that sets the return code to 16. For further information on ICHDEX01, see "Password authentication exits" on page 276.

Each time a user logs on and enters a password, password phrase, 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, password phrase, or OIDCARD stored on the database. If there is no match, the second step is performed.
2. RACF compares the results of the masking algorithm to the encoded form of the password, password phrase, or OIDCARD stored on the database.

For further information on ICHDEX01, see "Password authentication exits" on page 276.

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 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 might 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.

DES without the two-step method of checking is used when RACF cannot find an ICHDEX01 exit in the link pack area, and KDFAES is not enabled.

Using the masking algorithm

Guideline: Use the KDFAES algorithm, because it provides better security than the masking algorithm.

To use the masking algorithm, write your own ICHDEX01 exit (in the link pack area) that sets the return code to 4, see "Password authentication exits" on page 276.

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 276.

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 accordingly. See "Password authentication exits" on page 276.

PassTicket authentication

A PassTicket provides an alternative to the RACF password. 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. See Using PassTickets in *z/OS Security Server RACF Security Administrator's Guide* for more information on PassTickets. See Generating and evaluating a PassTicket in *z/OS Security Server RACF Macros and Interfaces* for information on generating PassTickets.

Planning Considerations for enabling KDFAES

While the new algorithm can be activated using the SETROPTS command, consider the following before activating it.

- "Create a backup copy of your RACF database" on page 59
- "Apply OA43998 and OA43999 to all systems lower than z/OS V2R2 that share the RACF database" on page 59
- "Apply service to other affected products" on page 59
- "Determine if programs you have written are affected" on page 59
- "RACF exit considerations" on page 60
- "RACF download considerations" on page 61
- "Performance and space considerations" on page 61
- "Test the change" on page 62

Note: The new encryption algorithm uses the Central Processor Assist For Cryptographic Function (CPACF) to perform SHA-256 operations. When the CPACF is not available, SHA-256 is performed in software. ICSF is not required in order to enable KDFAES.

Create a backup copy of your RACF database

Creating a backup of the RACF database is recommended whenever significant changes are being made to RACF and the RACF database.

Apply OA43998 and OA43999 to all systems lower than z/OS V2R2 that share the RACF database

Make sure that the service is applied on all sharing systems that are lower than z/OS V2R2, and that all the ++HOLD documentation has been reviewed.

Apply service to other affected products

A fix category is an identifier used to group and associate PTFs to a particular category of software service. The following fix category identifies the group of fixes that are required to support KDFAES.

IBM.Function.RACF.PasswordEncryption
: Fixes for z/OS Security Server RACF to support a stronger password encryption algorithm, and fixes for other z/OS software to support this enhancement.

For more information on fix categories, see IBM Fix Category Values and Descriptions (www.ibm.com/systems/z/os/zos/features/smpe/fix-category.html).

Informational APAR II14765 contains pertinent information (e.g. restrictions) for other software products related to this support.

Determine if programs you have written are affected

Check programs you have written to ensure that they can tolerate the new function.

Certain types of applications are affected when the new algorithm is activated. Any application that authenticates a user or stores into the RACF database using clear text is not affected.

Applications that are affected are those that have dependencies on the DES format of the PASSWORD field in the RACF database. For example, if the password field is extracted (using ICHEINTY or RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT) and compared against a password that the application has

encrypted itself (possibly by using RACROUTE REQUEST=EXTRACT,TYPE=ENCRYPT), this no longer works when the password on the RACF database is encrypted under KDFAES.

Another example is an application that passes a pre-encrypted password to RACROUTE REQUEST=VERIFY/X with ENCRYPT=NO. If the application supplies a current password extracted from the current password field, it fails when the password on the RACF database is encrypted under KDFAES. If the application supplies a new password that is encrypted using RACROUTE, this might not work after KDFAES is activated. The encryption method is specified on RACROUTE REQUEST=EXTRACT,TYPE=ENCRYPT,ENCRYPT=(data-addr,method). If the method specified is DES, this continues to work. If the method specified is STDDES or HASH, this does not work. If the method is INST, and there is no ICHDEX01 exit active, or the exit directs RACF to use DES, then this continues to work. However, if the exit directs RACF to use the masking algorithm, or if the exit performs its own encryption algorithm, this does not work.

Any application that copies a password or manages the password history is affected when the password is encrypted using KDFAES. That application then requires awareness of the active encryption algorithm and must take into account new password extension fields that are used for a KDFAES password.

RACF exit considerations

Determine if the new function affects RACF exits, if present on your system.

ICHDEX01 – Password authentication exit

The ICHDEX01 exit can be used to implement an encryption algorithm, or to instruct RACF to use DES or masking. These three algorithms are collectively referred to as “legacy”, to differentiate them from KDFAES. If there is no ICHDEX01 exit, RACF uses DES while evaluating passwords. After KDFAES is activated, RACF continues to call ICHDEX01 to evaluate a legacy password, but will no longer honor masking. When a password is changed under KDFAES, ICHDEX01 is no longer called for that password.

The considerations for activating KDFAES depend on the legacy algorithm currently active.

DES

If you have no ICHDEX01 exit, or your ICHDEX01 exit directs RACF to use the DES algorithm (by setting return code 8), then there are no issues with activating KDFAES, or performing a KDFAES conversion.

Installation-defined encryption method

If you have an ICHDEX01 exit that performs its own encryption algorithm, it needs to stay active until all passwords have been changed under KDFAES and until all non-KDFAES history entries have been replaced in the history. Do not perform a KDFAES conversion. The conversion function assumes that the input password hash is created with DES. If you do perform a conversion, a user is no longer able to log on with their existing password, and an administrative password change is required. Also, history entries are not usable. (A user is able to reuse any installation-encrypted password value that is contained in history).

You can determine when no more legacy passwords, phrases, and history values exist by using fields unloaded by IRRDBU00.

Masking

Masking is only active for encrypting passwords when you have an ICHDEX01 exit that sets return code 4. Since masking is never used when KDFAES is activated, there is more work to do on your part. Do not perform a KDFAES conversion. The conversion function assumes that the input password hash is created with DES. If you do perform a conversion, a user can no longer log on with their existing password, and an administrative password change is required. Instead, either:

- Convert to DES before activating KDFAES. This would entail changing ICHDEX01 to set a return code of 16, which causes RACF to encrypt new passwords using DES, and to try masking during evaluation if the DES evaluation fails. The password expiration function can be used to force users to change their password at their next log on attempt, however, this does not address existing history entries. It might not be easy to know when all user passwords have changed, as

RACF provides no way to differentiate a masked password from a DES password on the RACF database.

- Implement a bulk password change on a test system, and sacrifice your history entries. Follow these steps:

  1. Make a copy of the your RACF database.
  2. Activate this copy on a test system.
  3. Activate KDFAES on the test system.
  4. Perform a bulk password change, notifying users of their pending new password.
  5. Activate this copy on your production system.
  6. Remove your ICHDEX01 exit

  Attention: This results in unusable history entries (a user can reuse any previously masked password value contained in history).

ICHPWX01 – New password exit

The ICHPWXP parameter list contains the address to a password history structure (the PWXPWHST field). This parameter is passed to the exit by RACROUTE REQUEST=VERIFY/X and the PASSWORD command. When KDFAES is enabled, the PWXPWHST field is always 0, even if SETROPTS PASSWORD(HISTORY(n)) is active.

ICHPWX11 – New password phrase exit

When KDFAES is not active, the ICHPWX11 exit must approve a password phrase value between 9 and 13 characters, inclusive. In the absence of an ICHPWX11 exit, RACF rejects any password phrase less than 14 characters in length. A REXX-based sample exit is provided in SYS1.SAMPLIB.

When KDFAES is active, RACF allows a password phrase value of 9 characters or higher without an ICHPWX11 exit being active. You can remove the ICHPWX11 exit after enabling KDFAES if the only purpose of the exit is to allow a password phrase less than 14 characters.

If you use the sample exit for the additional checking it provides, and the version have installed is from a release lower than z/OS V2R2, note that the IRRPHREX exec requires a change if you plan to enable the new encryption algorithm and take advantage of the 9-character minimum password phrase length. The exit, as written, fails a password phrase less than 14 characters in length.

RACF download considerations

If you are using RACF downloads, determine if they are affected. The following downloads are available on the RACF download page (github.com/IBM/IBM-Z-zOS/tree/master/zOS-RACF/Downloads):

CUTPWHIS

This is a utility that removes extraneous passwords from the RACF password history. This function is now provided by the PWCLEAN keyword of the ALTUSER command.

Attention: Continued use of CUTPWHIS after activating the new encryption algorithm results in damaged password history and allows users to reuse password values.

PWDCOPY

This is a utility that copies passwords from one RACF database to another RACF database. Make sure you are using the updated version.

Attention: Continued use of the old version of PWDCOPY after activating the new encryption algorithm results in users being unable to log on on the target system of the copy.

Performance and space considerations

Consider two different aspects regarding performance, described below:

1. An increase in CPU required to evaluate and change passwords.
2. Fragmentation of the RACF database that can reduce performance of RACF database look-ups.

CPU consumption

The new encryption algorithm intentionally uses a higher amount of CPU than DES. It can be difficult to predict the effect this algorithm has on user response time and overall system performance, as it depends on the frequency of password evaluations.

This also affects password and password phrase changes, especially when history is active. Beyond the encryption that is required to verify the current value, the new value must also be encrypted. When history is active, the new value needs to be encrypted separately against each KDFAES history value to perform the comparison. If you have a high history value, and your users all tend to change their passwords on the same day, consider using the EXPIRED operand of the ALTUSER command to evenly distribute password expiration dates among your user population. Also, consider using the PWCLEAN function of ALTUSER if you have lowered your history value in the past.

Attention: IBM strongly recommends that you implement RACF ACEE caching using the IRRACEE class in VLF. For more information, see "ACEEs and VLF considerations" on page 73. For information on VLF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

RACF database performance and space utilization

Passwords, password phrases, and history encrypted under KDFAES occupies more space in USER profiles than they do under DES. A password extension field is defined in the RACF database templates for each of these entities to contain the additional information that does not fit within the current field. For password and password phrase history, a parallel field contains the extension information, and requires a corresponding set of generation numbers by which to index the elements. The size of a password, password history entry, and password phrase history entry increases by approximately 40 bytes. The current password phrase field is variable length. Under KDFAES, a password phrase is stored in the same manner as a password. Using your SETROPTS PASSWORD(HISTORY(n)) value, you can determine approximately how much more space is required. Ensure that your RACF database contains enough unused space to accommodate this, and increase the size of your database if necessary.

As passwords and password phrases change over time, and a user's profile grows in size, it might need to be split across data blocks in the RACF database. When this happens, references to this profile require additional I/O. IRRUT400 can be used to reorganize databases to bring the blocks back within close proximity of each other, including the index block that references them.

Consider performing a KDFAES conversion (for passwords and password history) up front to get the user profiles to their maximum size, and then perform a reorganization. Keep in mind, however, that conversion does not affect password phrases. If you use password phrases, the profile grows slowly over time, as they are changed.

Test the change

After activating KDFAES and changing your password, you can then log on to all applications that provide an authentication dialog. Then, change your password or password phrase with all such applications and verify that you can authenticate with the new value.

Using test workloads that are indicative of your production workloads, you can estimate performance impacts. You can consider performing a KDFAES conversion to ensure that your database can accommodate the extra space utilization. You could also use IRRUT400 to reorganize the database, and then activate that copy.

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 6. 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

Table 6. Format of ICHRSMFI (continued)

Name	Offsets DEC(HEX)	Length	Description	Format	Default
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. Can 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
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 127.

Chapter 4. Operating considerations

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

Enabling and disabling RACF

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

Enabling RACF

RACF is a component of the Security Server. Before you can use RACF on z/OS, the Security Server 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 the Security Server. 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 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(5650-ZOS)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(ENABLED)
```

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 for the Security Server to:

```
STATE(DISABLED)
```

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

For example, update the IFAPRD*xx* entry to look like this:

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        FEATURENAME('Security Server')
        ID(5650-ZOS)
        VERSION(*)
        RELEASE(*)
        MOD(*)
        STATE(DISABLED)
```

Dynamic parse and IRRDPI00

TSO parse macros can parse command keywords related to the base segments of profiles, but they cannot parse command keywords for other segments such as TSO, DFP, or OMVS. RACF provides the *dynamic parse* function to parse keywords for non-base segments. IRRDPI00 builds the dynamic parse table from the dynamic-parse specification data (IRRDPSDS) and profiles in the CFIELD class, and then starts dynamic parse.

Rules: To ensure that commands work properly, follow these rules:

© Copyright IBM Corp. 1994, 2020

- Run IRRDPI00 at every IPL. Otherwise, commands that allow information for non-base segments will not work properly.
- Do not change the dynamic-parse specification data (IRRDPSDS).

Guidelines: For best performance and use, follow these guidelines:

- Automate IRRDPI00, because you must run it after every IPL. For information about automating IRRDPI00, see "Automating IRRDPI00" on page 72.
- To add installation-defined keywords to the CSDATA segment, create profiles in the CFIELD class. Refer to *z/OS Security Server RACF Security Administrator's Guide* for more information.

If dynamic parse is not active, commands that refer only to RACF base segment information will work if they do not contain any typing mistakes. If you make a mistake, RACF attempts to invoke dynamic parse, and issues a message saying that dynamic parse is not active.

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* you run IRRDPI00. 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 "RACF database initialization utility program (IRRMIN00)" on page 209. After you update your templates, run IRRDPI00 on each system if the PTF also changed the dynamic parse specifications. Each time that 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 11 on page 156 shows a sample of the SET LIST output.

If the CFIELD class is active, the IRRDPI00 command processes CFIELD profiles. CFIELD profiles define custom field names and attributes in their CFDEF segments. When UPDATE is specified, the IRRDPI00 command stores the custom field keyword and attributes in the dynamic parse table. Validity checking of the fields will occur for both CHECK and UPDATE processing.

If a large number of custom fields are defined in the CFIELD class, ensure that the region size used to run IRRDPI00 is large enough to handle the increased storage usage.

Syntax of the IRRDPI00 command

```
IRRDPI00   {CHECK | LIST [(profile-type [segment-name [keyword-name]])] | UPDATE}
```

where:

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

If the CFIELD class is active, validity checking of the custom field definitions in the CFIELD class is performed. Fields with the following errors will result in an error message, and will not be included in the dynamic parse table:

- TYPE is not NUM, CHAR, HEX or FLAG
- MAXVALUE is less than MINVALUE for TYPE(NUM)
- Missing MAXVALUE for TYPE(NUM)
- Missing MINVALUE for TYPE(NUM)
- Value of FIRST is not correct for TYPE(NUM), TYPE(HEX), or TYPE(FLAG)
- Value of OTHER is not correct for TYPE(NUM), TYPE(HEX), or TYPE(FLAG)
- MAXLENGTH is missing or is not correct for the TYPE
- Values other than NOMINVALUE and NOMAXVALUE are set for TYPE(CHAR), TYPE(HEX), or TYPE(FLAG)

- MIXED(NO) is not set for TYPE(NUM), TYPE(HEX), and TYPE(FLAG)
- MAXVALUE/MINVALUE has more digits than is allowed by MAXLENGTH for TYPE(NUM)

LIST [(*profile-type* [*segment-name* [*keyword-name*]])]

Lists dynamic parse specification data. If you issue IRRDIP00 with no operands, LIST is the default. You can optionally specify for which profile type, segment, and keyword you want to list dynamic parse specification data. Note that if you do not specify a profile type and segment name, the output is quite large.

profile-type

The type of profile for which you want to list dynamic parse specification data. It can have the following values:

DATASET
DATASET profiles

GENERAL
General resource profiles

GROUP
GROUP profiles

USER
USER profiles

For example, to list dynamic parse specification data for all segments and keywords in the USER profile, issue:

```
IRRDPI00 LIST(USER)
```

segment-name

The name of the segment for which you want to list dynamic parse specification data, within the type of profile specified in *profile-type*. To specify *segment-name*, you must also specify *profile-type*. For a list of possible values for *segment-name*, see Table 7 on page 70.

For example, to list dynamic parse specification data for all fields in the TSO segment of the USER profile, issue:

```
IRRDPI00 LIST(USER TSO)
```

To list all custom fields currently in effect for USER profiles, issue:

```
IRRDPI00 LIST(USER CSDATA)
```

To list all custom fields currently in effect for GROUP profiles, issue:

```
IRRDPI00 LIST(GROUP CSDATA)
```

keyword-name

The name of the keyword for which you want to list dynamic parse specification data, within the segment specified in *segment-name* and the profile specified in *profile-type*. To specify *keyword-name*, you must also specify *profile-type* and *segment-name*. The keywords are from the commands used to define the segments. Table 7 on page 70 lists the profile types and segment names that you can specify, and identifies the commands that define them. To find the values of *keyword-name* that are valid for a combination of profile type and segment, look up the command in *z/OS Security Server RACF Command Language Reference* and find the keyword for the segment name. The subkeywords supported for the segment keyword are valid values for *keyword-name* on IRRDPI00.

For example, the TSO segment for the USER profile is defined by the ADDUSER command. The ADDUSER command allows you to specify keywords including ACCTNUM, COMMAND, DEST, and HOLDCLASS on the TSO keyword to define the TSO segment. To list the dynamic parse specification data for the ACCTNUM keyword, specify:

```
IRRDPI00 LIST(USER TSO ACCTNUM)
```

For the CSDATA segment of USER and GROUP profiles, valid values for *keyword-name* will be installation specific. This is because the CSDATA segment holds custom field data which can be defined by an installation using CFDEF segments in CFIELD profiles. For example, if your installation has defined the custom field keyword EMPSER for the CSDATA segment of the USER profile, you would enter the following to list the dynamic parse specification data for this keyword.

```
IRRDPI00 LIST(USER CSDATA EMPSER)
```

Contact your security administrator for information on the valid *keyword-name* values for the CSDATA segment of USER and GROUP profiles at your installation. Alternatively, you can omit the *keyword-name* to list all valid values for the CSDATA segment. For more information on custom fields and how to define them, refer to the *z/OS Security Server RACF Security Administrator's Guide*.

Table 7. Valid profile types and segment names for IRRDPI00

Profile type	**Segment name**	**Command that defines the segment**
DATASET	DFP	ADDSD
DATASET	TME	ADDSD
GENERAL	CDTINFO	RDEFINE
GENERAL	CFDEF	RDEFINE
GENERAL	DLFDATA	RDEFINE
GENERAL	EIM	RDEFINE
GENERAL	ICSF	RDEFINE
GENERAL	ICTX	RDEFINE
GENERAL	KERB	RDEFINE
GENERAL	PROXY	RDEFINE
GENERAL	SESSION	RDEFINE
GENERAL	SIGVER	RDEFINE
GENERAL	SSIGNON	RDEFINE
GENERAL	STDATA	RDEFINE
GENERAL	SVFMR	RDEFINE
GENERAL	TME	RDEFINE
GROUP	CSDATA	ADDGROUP
GROUP	DFP	ADDGROUP
GROUP	OMVS	ADDGROUP
GROUP	OVM	ADDGROUP
GROUP	TME	ADDGROUP
USER	CICS	ADDUSER
USER	CSDATA	ADDUSER
USER	DCE	ADDUSER

Table 7. Valid profile types and segment names for IRRDPI00 (continued)		
Profile type	**Segment name**	**Command that defines the segment**
USER	DFP	ADDUSER
USER	EIM	ADDUSER
USER	KERB	ADDUSER
USER	LANGUAGE	ADDUSER
USER	LNOTES	ADDUSER
USER	NDS	ADDUSER
USER	NETVIEW	ADDUSER
USER	OMVS	ADDUSER
USER	OPERPARM	ADDUSER
USER	OVM	ADDUSER
USER	PROXY	ADDUSER
USER	TSO	ADDUSER
USER	WORKATTR	ADDUSER

UPDATE
Performs syntax checks on the input data set and updates the dynamic parse table if no errors were found. If the CFIELD class is active, validity checking is performed on the CFIELD profiles, and the custom field keywords with no errors and their attributes are stored in the dynamic parse table.

IRRDPI00 errors and return codes

If IRRDPI00 fails with a nonzero return code, check SYSOUT output for the error message. See IRRDPI00 message descriptions in *z/OS Security Server RACF Messages and Codes*.

RACF authorization of the IRRDPI00 command

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

- Be given READ access to the IRRDPI00 resource in the FACILITY class

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

- Be given access to IRRDPI00 using RACF program control

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

- Be defined as a RACF SPECIAL user

If WHEN(PROGRAM) is in effect, you get an abend if you do not have READ access to PROGRAM IRRDPI00. None of the other authorization options listed on the existing page apply in this case.

If NOWHEN(PROGRAM) is in effect, you will receive an error message if you do not meet the FACILITY or SPECIAL access requirements documented. READ access to PROGRAM does not help in this case.

TSO/E authorization of the IRRDPI00 command

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

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 184 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 of the IRRDPI00 command" on page 71 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=2M,
//              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 105 for more information.
4. Authorize the RACF user ID associated with the IRRDPI00 started procedure. See "RACF authorization of the IRRDPI00 command" on page 71 for details.
5. Add IRRDPI00 to the TSO/E APF-authorized command table. See "TSO/E authorization of the IRRDPI00 command" on page 71 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 253. 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.

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 SETROPTS ADSP, SETPROPTS NOADSP, defining entire groups or updates from another system (z/VM or z/OS) 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, 2, and 3, 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 35.

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                          */
```

VLF considerations for program signature verification

RACF can use VLF to cache signature verification data in order to improve the performance of signature verification of signed program objects. This in turn can improve the load time of signed program objects. This is a consideration only for installations that choose to exploit signature verification.

For background information, see "Program signing and signature verification" on page 33. For implementation details, see Enabling RACF to verify signed programs in *z/OS Security Server RACF Security Administrator's Guide*.

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 signature verification data. Update the COFVLFxx member of SYS1.PARMLIB as follows:

```
CLASS NAME(IRRSPS0)         /* Signature Verification Data in Memory */
EMAJ(PERFCACHE)             /* Major name = PERFCACHE                */
```

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 127.
- 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 86.

- R_admin (IRRSEQ00) callable service

  When the RACF subsystem is active, it executes commands that are passed to it by R_admin. Applications that use R_admin 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.

- Key generation for the Network Authentication Server (IBM Kerberos)

  When a user profile has a KERB segment containing a Kerberos principal name (KERBNAME field) and the user sets a non-expired password, a key is generated and stored in the KERB segment of that user. When the change is because of an application update (for example, TSO or CICS logon), the RACF subsystem generates the key. If the RACF subsystem is not available, no key generation is performed for the password change.

- Password and password phrase enveloping

  When the password or password phrase enveloping function is configured, the RACF subsystem creates encrypted envelopes for eligible users when their passwords or password phrases are changed, and controls the retrieval of these envelopes by authorized applications. For details on the enveloping function, see *z/OS Security Server RACF Security Administrator's Guide*.

  When the enveloping function is configured, during RACF subsystem initialization RACF invokes z/OS UNIX services to initialize itself as a UNIX process, which requires the OMVS kernel to be initialized. If the OMVS kernel is not initialized, RACF subsystem initialization waits for OMVS initialization to complete. As a result, the RACF subsystem address space might initialize later in the IPL sequence than it would if enveloping was not configured.

  When the enveloping function is configured, an OMVS shutdown can affect the RACF subsystem. Enveloping operations wait for OMVS to be restarted. If enough password or password phrase changes are made while the OMVS kernel is unavailable, the available tasks in the RACF subsystem can be exhausted, affecting other RACF address space functions that would otherwise not be affected by an OMVS shutdown. An OMVS shutdown should not be performed while work is occurring on the system. For information about shutting down OMVS, see *z/OS UNIX System Services Planning*.

- LDAP event notification

  When LDAP event notification is configured, the RACF subsystem contacts the z/OS LDAP server to create a change log entry when a RACF user profile is updated. For more information, see *z/OS Security Server RACF Security Administrator's Guide*.

Tip: If you activate the RACF subsystem, perform the following tasks to allow the security administrator and system operator to address early startup issues because of RACF access failures:

1. Define the RACF subsystem as trusted. (For information about how to do this, see "Assigning a user ID to the RACF subsystem" on page 80.)
2. Ensure that the security administrator and the system operator know the correct command prefix for the RACF subsystem. (For more information, see "Updating the IEFSSNxx member of SYS1.PARMLIB" on page 78.)
3. Ensure that the security administrator has authorization to use the MVS LOGON operator command.

Taking these steps allows the security administrator and system operator to log on to an MVS console and repair RACF profiles.

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.

The RACF ASID is not reusable. Do not specify REUSASID=YES on the START command for the address space.

Activating the RACF subsystem

To activate the RACF subsystem, you must do the following:

- Update the IEFSSN*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 5 on page 78 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.

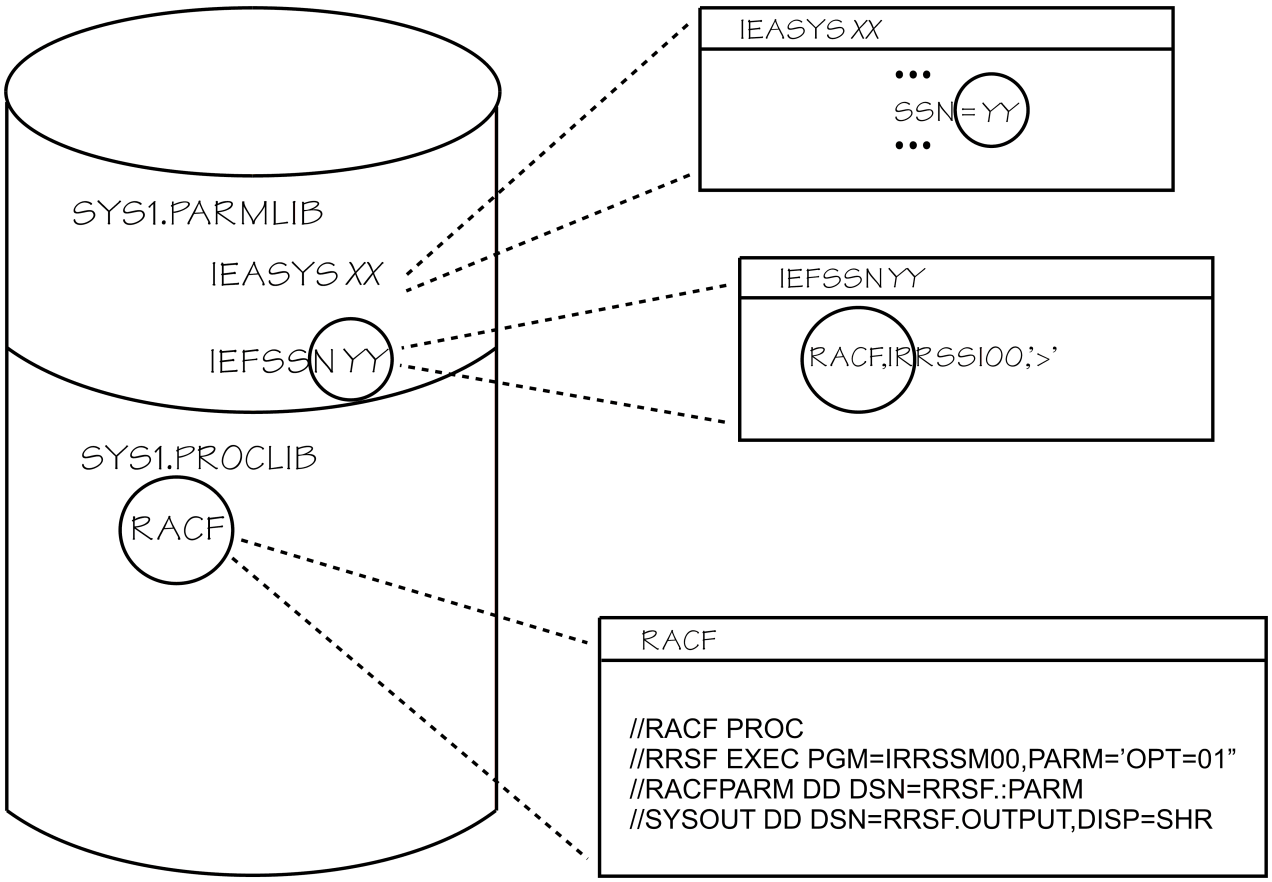


Figure 5. Setting up the RACF subsystem address space

Updating the IEFSSNxx member of SYS1.PARMLIB

The IEFSSN*xx* member of SYS1.PARMLIB 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, which is used in issuing RACF operator commands, and an optional command prefix scope. See *z/OS Security Server RACF Command Language Reference* for information about how to use the subsystem command prefix.

Rule: Place the entry for RACF after the entries for SMS and the primary subsystem (JES2 or JES3) in the IEFSSN*xx* member. For more information about the IEFSSN*xx* member of SYS1.PARMLIB, see *z/OS MVS Initialization and Tuning Reference*.

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 about CPF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Guideline: 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) INITRTN(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
```

Guideline: 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, which is enclosed in parentheses, instead of a command prefix.

If you do not specify a scope, the quotation marks 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 about the DISPLAY OPDATA command.

If you do not specify a scope, the command prefix is not registered with CPF. 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 by way of the MVS ROUTE command).

Guideline: Specify a scope.

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 82). 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.

Example: 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.

Example: 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.

Example: 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, quotation marks are optional on the command prefix.

Example: 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.

Example: 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 by way of the MVS ROUTE command). Because a scope is specified, the command prefix is registered with CPF.

Example: 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.

Note:

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 and 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*. You must ensure that the assigned user ID has sufficient authority to issue any commands defined in an IRROPTxx RACF parameter library member. For more information on the RACF parameter library, see "The RACF parameter library" on page 184.

Guideline: Assign a *protected user ID* to the RACF subsystem. A user ID becomes a protected user ID when it is assigned the NOPASSWORD, NOPHRASE, and NOOIDCARD attributes by an ADDUSER or

ALTUSER command. A protected user ID cannot be revoked due to incorrect password or password phrase attempts or used to enter the system in ways that require a password or password phrase. 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.

The security administrator can assign a RACF user ID to the RACF subsystem using the STARTED class. If your installation has not activated the STARTED class, you can use the started procedures table (ICHRIN03). For more information, see "Associating started procedures and jobs with user IDs" on page 105.

Example: The following example shows how you could assign a RACF user ID to the RACF subsystem using ICHRIN03.

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'40'          Trusted
RESERVED   DC     XL7'00000000000000'
           END
```

Guideline: Define the RACF subsystem as *privileged* or *trusted*. You can use either the STARTED class or ICHRIN03 to do this.

Example: If your installation has activated the STARTED class, the security administrator can create a profile for the RACF subsystem in the STARTED class marked trusted, as shown in the following example, where RACF is the name of the subsystem and RACFAS is the RACF-defined user ID:

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

Example: If you are using ICHRIN03 instead of the STARTED class, the following example illustrates how to assign the RACF user ID SUBSYS to the RACF subsystem and mark it trusted:

```
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'40'          Entry is trusted
RESERVED   DC     XL7'00000000000000'
           END
```

Additional setup for the RACF subsystem user ID

Some functions require additional setup for the RACF subsystem user ID.

- If you are using the Network Authentication Server (IBM Kerberos), the subsystem user ID must have a z/OS UNIX UID, and its default group must have a z/OS UNIX GID. After the security administrator adds the OMVS segments, restart the RACF subsystem.
- If you are using the RACF remote sharing facility (RRSF) with the TCP/IP protocol, the subsystem user ID must have a z/OS UNIX UID, and its default group must have a z/OS UNIX GID. After the security administrator adds the OMVS segments, make the local node operative again. (You do not have to restart the RACF subsystem.)
- If your installation plans to enable enveloping for passwords and password phrases, the subsystem user ID must have a z/OS UNIX UID, and its default group must have a z/OS UNIX GID. In addition, the security administrator must give the user ID READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class. For more information, see *z/OS Security Server RACF Security Administrator's Guide*. After the security administrator adds the OMVS segments and activates the enveloping function (that is,

defines the PASSWORD.ENVELOPE or PASSPHRASE.ENVELOPE resource and activates the RACFEVNT class), restart the RACF subsystem.

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 that the name of the procedure matches your IEFSSN*xx* entry. Here is an example:

```
//RACF PROC
//RACF EXEC PGM=IRRSSM00,REGION=0M
//SYSOUT DD DSN=RRSF.OUTPUT,DISP=SHR
```

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 configure a SYSOUT DD include a SYSOUT DD statement specifying the name of the pre-allocated dataset. Here is an example:

```
//RACF PROC
//RRSF EXEC PGM=IRRSSM00,REGION=0M,PARM=OPT=01
//RACFPARM DD DSN=RRSF.PARM,DISP=SHR
//SYSOUT DD DSN=RRSF.OUTPUT,DISP=SHR
```

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 do not specify it on the PARM='OPT=*xx*' parameter.
- If you do not have an IRROPT00 member, and do not specify a PARM='OPT=*xx*' parameter, RACF tries to find an IRROPT00 member during initialization, and issues a warning that it could not find one.

If you use a RACFPARM DD statement, make sure that you do not specify FREE=CLOSE. As a result, IEC988I is issued. An example of that message indicates:

```
IEC988I jjj,sss,ddn{-#},dev,volser,dsn DATA SET NOT UNALLOCATED DURING
             CLOSE RCxx
01      Disposition of LEAVE specified.
```

See *z/OS MVS System Messages, Vol 7 (IEB-IEE)* for more information about message IEC988I.

The following JCL activates the RACF subsystem and automatically processes the member IRROPT01 in the RACF parameter library that is 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 about the RACF parameter library, see "The RACF parameter library" on page 184.

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 85.)

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 xxxxx,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 xxxxx,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.

The RACF ASID is not reusable. Do not specify REUSASID=YES on the START command.

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. For the syntax of the RESTART command, see *z/OS Security Server RACF Command Language Reference*. For information on security for the RESTART command, see *z/OS Security Server RACF Security Administrator's Guide*.

The functions you can specify on the RESTART command are shown in Table 8 on page 83.

Table 8. Functions you can restart using the RESTART command

Function Keyword	Function Restarted	Modules Reloaded
COMMAND	Command handler	IRRSSC00

Table 8. Functions you can restart using the RESTART command (continued)		
Function Keyword	**Function Restarted**	**Modules Reloaded**
CONNECTION	Local node and APPC protocol manager task	IRRDDM00
	APPC inbound communication subtask	IRRAPPC0
	APPC outbound communication subtask	IRRAPPC2
	APPC handshaking subtask	IRRAPPC6
	Local mode communication subtask	IRRSSL00
	TCP protocol manager task	IRRTCP00
	TCP connector subtask	IRRTCP01
	TCP handshaking and communication subtask	IRRTCP02
MESSAGE	Message processor	IRRSSMG0
OUTPUT	Output handler	IRRSSOP0
RACLINK	RACLINK task	IRRSSK00
RECEIVE	RRSF request receiver	IRRSSR00
SEND	RRSF request sender	IRRSSND0
SIGNAL	XCF signal task	IRRSIG00

The RESTART CONNECTION NODE(*nodename*) command can be used to restart a pair of handshaking and communication subtasks. The NODE keyword specifies the remote node for which the subtask pairs are to be restarted. You can specify NODE(*) to restart all handshaking and communication subtasks.

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 terminates 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. For the syntax of the RACF STOP command, see *z/OS Security Server RACF Command Language Reference*. For information on security for the RACF STOP command, see *z/OS Security Server RACF Security Administrator's Guide*. (Note that there is also an MVS STOP command. The RACF STOP command and the MVS STOP command are not related.)

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 things:

- 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 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 127.

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 z/OS 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*.

Tip: To determine whether the database is on a device that has been configured as shared, issue an RVARY LIST command. If the device is not shared, the output includes a column labeled SHR, with the value N. The column does not appear if the device is shared.

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 the MVS global resource serialization function to convert RESERVES, all z/OS systems that access the RACF database must be part of the same global resource serialization complex, and there can be no z/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 the MVS global resource serialization function.

If your shared RACF database is at application identity mapping (AIM) level 1 or higher, all systems that update the OMVS segment of USER or GROUP profiles, or update the ALIAS segment of general resource profiles (for example, any SERVAUTH class profile), or run RACF utilities, should have global resource serialization connections between the systems, should be in the same global resource serialization complex, and should be running OS/390 release 10 or any z/OS release. Adding or deleting a profile that has any of these segments, altering these segments, or running RACF utilities from a system outside the global resource serialization complex might result in incorrect results; for example, an alias index entry for an OMVS UID or SERVAUTH alias might point to the wrong profile, or to one that does not exist. To prevent database sharing errors, it might be useful to use RACF program control to restrict access to all RACF commands that can update these segments, to ensure that they cannot be used from systems outside a single global resource serialization complex.

If you do get your alias index out of synchronization with the USER or general resource profiles, you might need to delete and re-create some profiles or alter some data (for example, a UID or GID), in order to correct the inconsistency. For more information, see “Recovering from errors with application identity mapping” on page 324.

Using the global resource serialization function

A global resource serialization complex lets users on multiple z/OS 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 9. RACF ENQ resources

Major	Minor	Type	Notes
ICHRGL01	SIGNON.ENQ	STEP	
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 the MVS global resource serialization function, 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 hexadecimal address. *bbbb* is 4 blank characters.
SYSZRACF	ACEE3PTY*@@@@CGRP	STEP	@@@@ is a hexadecimal address.
SYSZRACF	AHSTABLE	STEP	
SYSZRACF	AHSTUSER*userid*####	STEP	*userid* is the caller's user ID. #### is a 4-byte EBCDIC value consisting of printable characters 0-9, and A-F (X'F0'-X'F9' and X'C1'-X'C6').
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 9. 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. If RACF is not enabled for sysplex communication, RACF issues this as a SYSTEM ENQ. Your global resource serialization product, such as the MVS global resource serialization function, 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. If RACF is enabled for sysplex communication, RACF issues this as a SYSTEMS ENQ. Your global resource serialization product, such as the MVS global resource serialization function, must treat this as a multisystem ENQ.
SYSZRACF	DSDTPREP...DSDTPREP	SYSTEMS	The minor name consists of 6 occurrences of "DSDTPREP" concatenated into one string.
SYSZRACF	MAINSWITCHINPROGRESS	SYSTEMS	Dynamic MAIN switch in progress using the TARGET NEWMAIN or TARGET PLEXNEWMAIN command.
SYSZRACP	*dsn*	SYSTEMS	*dsn* is any data set name. RACF issues this as a RESERVE; however, your global resource serialization product, such as the MVS global resource serialization function, 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	IF RACF is not enabled for sysplex communication, RACF issues this as a SYSTEM ENQ. Your global resource serialization product, such as the MVS global resource serialization function, must treat this as a single system ENQ.

Table 9. RACF ENQ resources (continued)

Major	Minor	Type	Notes
SYSZRAC2	IRRCRV05	SYSTEMS	If RACF is enabled for sysplex communication, RACF issues this as a SYSTEMS ENQ. Your global resource serialization product, such as the MVS global resource serialization function, 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	SSTABLE1	SYSTEM	
SYSZRAC2	SSTABLE2	SYSTEM	
SYSZRAC2	XMCAXMCA...XMCA	SYSTEM	The minor name consists of 12 occurrences of "XMCA" concatenated into one string.
SYSZRAC2	GLOBALGLOBALGLOBAL	SYSTEMS	
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.
SYSZRAC2	TEMPLATD	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	

Table 9. RACF ENQ resources (continued)			
Major	**Minor**	**Type**	**Notes**
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	00ICTX*bbbbbbbbbbbb*CACHE*bbb*	SYSTEM	Obtained and released during a store request that creates a read/write cache managed by R_cacheserv to ensure that only one cache is created. Also obtained and released during a cache destroy request for a read/write cache. *bbbbbbbbbbbb* is 12 blank characters. *bbb* is 3 blank characters.
SYSZRAC8	00*cachenamebbbbbbbbbb*DASPHYDS	SYSTEM	*cachename* is the name of an R_cacheserv managed cache. *bbbbbbbbbb* is 10 blank characters.
SYSZRAC9	DYNCDT	SYSTEM	Obtained to serialize updates to the dynamic class descriptor table.

Sysplex considerations

The z/OS 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 z/OS 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, all systems in a sysplex should 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 z/OS 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 documents are valuable sources for learning more about a sysplex and the coupling facility:

- *z/OS Security Server RACF Diagnosis Guide*
- *z/OS MVS Setting Up a Sysplex*

Sharing a database

See "Shared database considerations" on page 87 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 is compatible on all sharing systems.
- The database range table is identical on all sharing systems.
- The class descriptor table (ICHRRCDE) is compatible on all sharing systems.

z/OS and z/VM systems can share a RACF database. 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 93), nor outside of a single global resource serialization complex if you choose to convert the SYSZRACF RESERVE to an ENQ using the z/OS 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 performs command propagation for certain commands (for example SETROPTS RACLIST). You do not 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 100.

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:

- 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 the MVS 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 SYSZRAC2 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 this name dynamically, because RACF maintains a permanent ENQ on this resource.
  - If you have SYSZRACF in your RNL, you can remove it dynamically if you first stop the RACF subsystem on all systems in the global resource serialization complex. SYSZRACF (minor name of RACF) is held continuously if a RACF subsystem is running, and stopping the RACF subsystems releases the ENQ.
- 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 87, in non–data sharing mode hardware RESERVEs are used to serialize database access. If using the MVS 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 87.
- 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 is compatible on all sharing systems.
  - The database range table is identical on all sharing systems.
  - The class descriptor table (ICHRRCDE) is compatible on all sharing systems.

Guideline: All systems enabled for sysplex communication and sharing the same RACF database should 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 that 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 MVS 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 have z/OS Security Server RACF enabled.

  Note: If a z/OS 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 SYSZRAC2 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 this name dynamically, because RACF maintains a permanent ENQ on this resource.
  - If you have SYSZRACF in your RNL, you can remove it dynamically if you first stop the RACF subsystem on all systems in the global resource serialization complex. SYSZRACF (minor name of RACF) is held continuously if a RACF subsystem is running, and stopping the RACF subsystems releases the ENQ.
- 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 is compatible on all sharing systems.
- The database range table is identical on all sharing systems.
- The class descriptor table (ICHRRCDE) is compatible on all sharing systems.

Attention:

- If any system is using the database in data sharing mode, then all systems accessing the database must be in the same sysplex and must use the database in data sharing mode. If you attempt to share the database with a system (z/OS or VM) outside of the sysplex, or if a system in the sysplex attempts to use the database in non-data sharing mode, database corruption will occur.
- If RACF detects possible incorrect use of the database that would result in data corruption, it issues a WTOR message to the system operator, prompting the operator to verify that the database is not being used incorrectly. For details about how to respond to any of these messages, refer to *z/OS Security Server RACF Messages and Codes*.

If you have 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 "Overview of the RRSF function" on page 127 for information on automatic direction.

Guarding against data corruption resulting from incorrect database sharing

When operating RACF in data sharing mode, a coupling facility is used to cache recently accessed RACF data, and access is serialized to that data to ensure its validity. Corruption will occur when a database is used in data sharing mode, but some system that is not using the coupling facility is also accessing the database. For this reason, if any system is using the database in data sharing mode, then all systems accessing the database must be in the same sysplex and must use the database in data sharing mode.

- If a database is being used in data sharing mode and an individual system accesses the same database in non-data sharing mode, data corruption will occur because the coupling facility will not be made aware of any updates made by the system that is in non-data sharing mode.
- If a database is in data sharing mode and systems in another sysplex attempt to access the same database, data corruption will occur regardless of the data access mode of the other sysplex. This is because a coupling facility can only span one sysplex.

To help guard against data corruption resulting from incorrect database sharing, systems running z/OS V1R10 or higher use profiles to track how the RACF database has been accessed. These profiles are named IRRPLEX_*sysplex-name* and are in the GXFACILI class. The APPLDATA field of an IRRPLEX_*sysplex-name* profile identifies the data access mode as either "DATA SHARING MODE" or "NON-DATA SHARING MODE". When you IPL a system, use the RVARY ACTIVE command to activate a database, or use the RVARY DATASHARE command to enter data sharing mode, these profiles help the system detect, and notify the system operator, if there are indications that a database is about to be used in a way that risks data corruption. WTOR messages enable the system operator to judge if the system is indeed about to use the database in a way that would cause data corruption, and can cancel the operation to avoid this.

If no IRRPLEX_*sysplex-name* profile exists for a sysplex at IPL time, the system creates one. The system will determine the initial mode (data sharing mode or non-data sharing mode) of the sysplex using the data sharing bits of the data set name table. Because IRRPLEX_*sysplex-name* profiles are created automatically during the first IPL of a z/OS V1R10 or higher system, you do not need to manually create them. However, by manually creating IRRPLEX_*sysplex-name* profiles, you can improve the ability of the system to warn the system operator about potential corruption.

Whether created automatically by the system at IPL time or created manually to improve to efficiency of the database corruption checks, when the data sharing mode of the sysplex is changed by the RVARY command, the system will update the IRRPLEX_*sysplex-name* profile to reflect this.

If you are using the RACF remote sharing facility (RRSF), keep in mind that changes to IRRPLEX_*sysplex-name* profiles in the GXFACILI class should not be propagated to remote nodes. When the system creates

or modifies an IRRPLEX_*sysplex-name* profile (at IPL-time or as a result of the mode being changed by the RVARY command), its actions are not propagated to remote nodes even if automatic direction of application updates is active. If you are manually creating or modifying an IRRPLEX_*sysplex-name* profile (using the RDEFINE, RALTER, and RDELETE commands), and automatic command direction is active for the GXFACILI class, be sure to use the ONLYAT keyword on the command to prevent the changes being propagated to other nodes.

Guarding against data corruption resulting from incorrect database sharing at IPL time

At IPL-time, the system uses the sysplex communication and data sharing bits of the data set name table to determine if RACF is enabled for sysplex communication, and whether it is to be used in data sharing mode or non-data sharing mode. If this is the first system in the sysplex to enter RACF sysplex communication and join the XCF group IRRXCF00, the mode specified in its data set name table is the initial mode of the sysplex communication group. If this is not the first system to join the IRRXCF00 group, it will IPL into the mode used by the rest of the systems in RACF sysplex communication.

Once the system determines the mode into which the system will IPL, it will locate all IRRPLEX_*sysplex-name* profiles in the GXFACILI class to determine how the database has been accessed previously. If the information provided by the IRRPLEX_*sysplex-name* profiles indicates that IPLing in the desired mode might risk data corruption, a WTOR message will be sent to the system console. For example:

Table 10. Potential data corruption messages displayed at IPL time

If IPLing into:	**And the APPLDATA field on:**	**Then a WTOR message is sent to the system console to inform the system operator that:**	**The system operator can choose to:**
Non-data sharing mode	Any IRRPLEX_*sysplex-name* profile indicates "DATA SHARING MODE"	A profile indicates the database was last used in data sharing mode, and database corruption will result if it is currently being used in data sharing mode and this system accesses it in non-data sharing mode.	• Continue IPLing into non-data sharing mode, if he or she is certain the database is not being used by another system in data sharing mode. • If the database is being used by a sysplex in data sharing mode, IPL the system into failsoft mode to avoid data corruption.

Table 10. Potential data corruption messages displayed at IPL time (continued)

If IPLing into:	And the APPLDATA field on:	Then a WTOR message is sent to the system console to inform the system operator that:	The system operator can choose to:
Data sharing mode	Any IRRPLEX_*sysplex-name* profile indicates "NON-DATA SHARING MODE"	A profile indicates the database was last used in non-data sharing mode, and database corruption will result if it is currently being used in non-data sharing mode and this system accesses it in data sharing mode.	• Continue IPLing into data sharing mode, if he or she is certain the database is not being used by another system in non-data sharing mode. • If other systems are using the database in non-data sharing mode, and this is the first member of the IRRXCF00 group, IPL into non-data sharing mode instead of data sharing mode. • If other systems are using the database in non-data sharing mode, and this is not the first system in the IRRXCF00 group, IPL the system into failsoft mode. (Because this is not the first system in the IRRXCF00 group, however, data corruption might have already occurred.)
	An IRRPLEX_*sysplex-name* profile for any other sysplex indicates "DATA SHARING MODE"	A profile indicates the database has been used by another sysplex in data sharing mode. Since all systems accessing the database in data sharing mode must be in the same sysplex, database corruption will result if it is currently being used by another sysplex.	• Continue IPLing into data sharing mode, if he or she is certain the database is not being used by another sysplex. • If the database in being used by another sysplex, but that sysplex is in non-data sharing mode and this is the first member of the IRRXCF00 group, IPL into non-data sharing mode instead of data sharing mode. • If the database is being used by another sysplex in data sharing mode, IPL the system into failsoft mode.

The warnings about possible data corruption at IPL time are only as reliable as the information provided in the IRRPLEX_*sysplex-name* profiles in the GXFACILI class. If you do not manually create IRRPLEX_*sysplex-name* profile(s) to describe how the RACF database should be accessed, the first system joining the sysplex will create the IRRPLEX_*sysplex-name* profile for that sysplex. This data sharing mode information for the sysplex is then available and will be checked during any subsequent IPL of a z/OS V1R10 or higher system, or when a V1R10 or higher system uses the RVARY command to either enter data sharing mode, or activate a data set of the database. However, until a sysplex member running z/OS V1R10 or higher is initialized, there will be no IRRPLEX_*sysplex-name* profile for that sysplex, and the system is therefore less capable to warn the operator about potential data corruption due to incorrect database sharing. By creating one or more IRRPLEX_*sysplex-name* profiles manually to fully describe how the RACF database is being accessed, you can increase the system's ability to detect situations that could lead to data corruption.

Note: Do not use the RACF remote sharing facility (RRSF) to propagate RDEFINE, RALTER, and RDELETE commands that manipulate IRRPLEX_*sysplex-name* profiles in the GXFACILI class to other databases. If automatic command direction is enabled for the GXFACILI class, and you are using the RDEFINE, RALTER, or RDELETE command to create, alter, or delete an IRRPLEX_*sysplex-name* profile, use the ONLYAT operand on the RDEFINE, RALTER, or RDELETE command to prevent propagation. The ONLYAT operand must be used whether you are manipulating the class GXFACILI IRRPLEX_*sysplex-name* profiles on a local or remote node.

The RDEFINE and RDELETE command examples that follow specify the ONLYAT operand. If RRSF is not being used to propagate RDEFINE, RALTER, and RDELETE commands on your system, then do not specify ONLYAT when updating the class GXFACILI IRRPLEX_*sysplex-name* profiles. If you are unsure whether RRSF is propagating these commands on your system, ask your security programmer or refer to Chapter 5, “RACF remote sharing facility (RRSF),” on page 127.

Table 11. Advantages of creating IRRPLEX_sysplex-name profiles manually

For example:	**You have the system automatically generate IRRPLEX_*sysplex-name* profile at IPL-time:**	**You have manually created one or more IRRPLEX_*sysplex-name* profiles to fully describe how the RACF database is being accessed:**
A sysplex, whose members are running z/OS V1R9 or lower, is accessing a database in data sharing mode. One system in that sysplex has z/OS V1R10 or higher installed and is IPLed. However, by mistake, the sysplex communication and data sharing bits were not set in the data set name table.	Because the sysplex communication and data sharing bits were not set in the data set name table, the system is initialized into non-data sharing mode. Because the database is already being used by a sysplex in data sharing mode, this will likely lead to data corruption.	Because the sysplex communication and data sharing bits were not set in the data set name table, the system determines that it will initialize into non-data sharing mode. However, before it initializes into this mode, it first checks for any IRRPLEX_*sysplex-name* profiles in the GXFACILI class. Because it finds a profile indicating that a sysplex might already be accessing the database in data sharing mode, it sends a WTOR message to the system operator. The system operator is able to avoid corruption of the database by IPLing the system into failsafe mode. The sysplex communication and data sharing bits can then be correctly set in the data set name table, and the system can be re-IPLed.
Two sysplexes are each using a RACF database in non-data sharing mode. The systems in one of the sysplexes are migrated to a z/OS V1R10 or higher release and IPLed. However, by mistake, the sysplex communication and data sharing bits in the data set name table request data sharing mode at IPL.	The sysplex is initialized into data sharing mode. Because another sysplex is accessing the database in non-data sharing mode, this will likely lead to data corruption.	Based on the sysplex communication and data sharing bits set in the data set name table, the system determines that it will initialize into data sharing mode. However, before it initializes into this mode, it first checks for any IRRPLEX_*sysplex-name* profiles in the GXFACILI class. Because it finds two profiles indicating that a sysplex might already be accessing the database in non-data sharing mode, it sends a WTOR message to the system operator. The system operator is able to avoid corruption of the database by IPLing into non-data sharing mode.

Before you IPL a z/OS V1R10 or higher system, you can improve the system's ability to detect incorrect database sharing by creating one or more IRRPLEX_*sysplex-name* profiles in the GXFACILI class to fully describe how the RACF database is being accessed. In the following command examples, you should replace *sysplex-name* with the actual name of the sysplex. To obtain the name of a sysplex, you can issue the following command:

```
D XCF,SYSPLEX
```

If the database is being accessed in data sharing mode, issue the following command to describe how the database is being accessed.

```
rdefine gxfacili irrplex_sysplex-name APPLDATA('DATA SHARING MODE') ONLYAT(.userid)
```

If the database is being accessed in non-data sharing mode, it is possible that any combination of one or more sysplexes, monoplexes, or systems in XCF-local mode could all be accessing the same database. While the RACF database can be shared among multiple sysplexes and systems in this way, data corruption will result if any sysplex should begin data sharing mode. To help prevent this, you should fully describe how the database is being used by creating an IRRPLEX_*sysplex-name* profile in the GXFACILI class for every sysplex, monoplex, and system in XCF-local mode that is accessing the database in non-

data sharing mode. Issue the following command for each sysplex and monoplex, and any system in XCF-local mode:

```
rdefine gxfacili irrplex_sysplex-name APPLDATA('NON-DATA SHARING MODE')
ONLYAT(.userid)
```

If you are copying a RACF database to be used in another environment, you should be aware that any IRRPLEX_*sysplex-name* profiles it contains might not be appropriate for the new environment. Copying the profiles can result in misleading warnings and otherwise compromise the reliability and efficiency of the potential database corruption checks when you IPL a system in the new environment. To avoid these problems, you can remove and define profiles as needed before copying the database using the IRRUT200 or IRRUT400 utility so that any IRRPLEX_*sysplex-name* profiles in the GXFACILI class reflect how the database will be used in the target environment. To list IRRPLEX_*sysplex-name* profiles in the GXFACILI class, issue the following command:

```
search class(gxfacili) mask(irrplex_)
```

To remove any of these profiles that will not be needed in the target environment, issue the RDELETE command:

```
rdelete gxfacili irrplex_sysplex-name ONLYAT(.userid)
```

To create additional profiles that will be needed in the target environment, issue the RDEFINE command. Use the APPLDATA field to indicate either "DATA SHARING MODE" or "NON-DATA SHARING MODE" as appropriate.

```
rdefine gxfacili irrplex_sysplex-name APPLDATA('DATA SHARING MODE') ONLYAT(.userid)
```

```
rdefine gxfacili irrplex_sysplex-name APPLDATA('NON-DATA SHARING MODE')
ONLYAT(.userid)
```

After copying the database, you can remove and redefine the profiles as needed for the current environment.

Guarding against data corruption resulting from incorrect database sharing when changing the status of a RACF database

When issuing the RVARY command on a z/OS V1R10 or higher system to activate a database (by specifying the ACTIVE parameter), or to enter data sharing mode (by specifying the DATASHARE parameter), the RVARY command will locate all IRRPLEX_*sysplex-name* profiles in the GXFACILI class to determine how the database has been accessed previously. The RVARY command checks the IRRPLEX_*sysplex-name* profiles in the database, because it could be a different database than the one checked at IPL-time. If the RVARY ACTIVE command is issued and the information provided by the IRRPLEX_*sysplex-name* profiles indicates that activating the database might risk data corruption, a WTOR message will be sent to the system console. For example:

Table 12. Potential data corruption messages displayed when issuing the RVARY ACTIVE command

If issuing the RVARY ACTIVE command to:	And the RVARY command finds:	Then a WTOR message will be sent to the system console to inform the system operator that:	The system operator can choose to:
Activate a database in non-data sharing mode	Any IRRPLEX_*sysplex-name* profile whose APPLDATA field indicates "DATA SHARING MODE"	A profile indicates the database was last used in data sharing mode, and database corruption will result if it is currently being used in data sharing mode and this system accesses it in non-data sharing mode.	• Continue the operation, if he or she is certain the database is not being used by another system in data sharing mode. • If the database is being used by a sysplex in data sharing mode, cancel the operation.

Table 12. Potential data corruption messages displayed when issuing the RVARY ACTIVE command (continued)

If issuing the RVARY ACTIVE command to:	And the RVARY command finds:	Then a WTOR message will be sent to the system console to inform the system operator that:	The system operator can choose to:
Activate a database in data sharing mode	Any IRRPLEX_*sysplex-name* profile besides the one for the sysplex.	A profile besides the one for the current sysplex was found, indicating that another sysplex might be using the database. If so, accessing the database in data sharing mode might result in database corruption.	• Continue the operation, if he or she is certain the database is not being used by another sysplex. • If another sysplex is using the database, cancel the operation.
	No IRRPLEX_*sysplex-name* profiles except the one for the current sysplex, and the APPLDATA field of the profile indicates "NON-DATA SHARING MODE"	A profile indicates the database was last accessed by the sysplex in non-data sharing mode. The sysplex is about to start accessing the database in data sharing mode.	• Continue the operation, if he or she is certain the database is not being used by any system outside the sysplex. • If the database is being used by any system outside the sysplex, cancel the operation.

If the RVARY DATASHARE command is issued and the information provided by the IRRPLEX_*sysplex-name* profiles indicates that entering data sharing mode might risk data corruption, a WTO message will be sent to the system console and the RVARY DATASHARE will be cancelled. Specifically, if issuing the RVARY DATASHARE command to enter data sharing mode, and the RVARY command finds any IRRPLEX_*sysplex-name* profile besides the one for the sysplex, then a WTO message will be sent to the system console. This WTO message will inform the system operator that a profile besides the one for the current sysplex was found, indicating that another sysplex might be using the database. If so, accessing the database in data sharing mode might result in database corruption. For this reason, the command does not complete successfully. After the system cancels the RVARY DATASHARE to avoid database corruption, you can check the profiles to determine if they accurately reflect the way the database is being accessed.

- If the profiles are correct, do not attempt to enter data sharing mode.
- If the profiles are extraneous and do not reflect the actual database usage, you can RDELETE them and reissue the RVARY DATASHARE command.

If the RVARY command does change the mode, it will also update the APPLDATA field of the sysplex's IRRPLEX_*sysplex-name* profile to reflect the new mode.

Note: Keep in mind that the RVARY command performs these additional database-protection checks only if the command is issued on a z/OS V1R10 or higher system. If you have a sysplex that combines systems running V1R10 or higher with systems running V1R9 or lower, you should issue the RVARY command from a V1R10 or higher system. If issued from a V1R9 or lower system in the sysplex, the RVARY command will not perform any database-protection checks.

Also be aware that, when issuing the RVARY command on a z/OS V1R10 or higher system to begin data sharing mode (by specifying the DATASHARE parameter) or non-data sharing mode (by specifying the NODATASHARE parameter), the RVARY command will update the APPLDATA field of the sysplex's IRRPLEX_*sysplex-name* profile to reflect the new mode. For this reason, you should also issue any RVARY DATASHARE or RVARY NODATASHARE command from a V1R10 or higher system. If issued from a V1R9 or lower system, the RVARY command will not update the APPLDATA field of the sysplex's IRRPLEX_*sysplex-name* profile to reflect the new mode. Because the APPLDATA field will no longer reflect the last mode the sysplex used to access the database, this will compromise the capability and reliability of the checks that are performed by the V1R10 systems in the sysplex.

The warnings about possible data corruption generated by the RVARY command are, like the warnings generated at IPL-time, only as reliable as the information provided in the IRRPLEX_*sysplex-name* profiles in the GXFACILI class.

"Guarding against data corruption resulting from incorrect database sharing at IPL time" on page 95 contains information on how you can improve the system's ability to detect incorrect database sharing by creating one or more IRRPLEX_*sysplex-name* profiles in the GXFACILI class to fully describe how the

RACF database is being accessed. You should take these actions before you IPL the R1V10 or higher system to improve the reliability of the checks performed at IPL-time. If, however, you did not take these actions prior to IPLing the system, you can still take them after the IPL to improve the reliability of the checks performed by the RVARY command.

Sysplex communication

If you are in the planning stages of configuring for a sysplex, see Parallel Sysplex (www.ibm.com/systems/z/advantages/pso) 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.

RACF support for the R_cacheserv SAF callable service (IRRSCH00) exploits RACF sysplex communication. When RACF is enabled for sysplex communication and RACF determines that an R_cacheserv retrieve or remove request (function code X'0006', options X'0003', X'0004', or X'0005') is for data that is cached on another member of the sysplex, RACF attempts to retrieve or remove the data from the other member. Use of an identity cache in a sysplex requires that RACF is enabled for sysplex communication.

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.)

When RACF is enabled for sysplex communication, and a command is propagated to other systems, it runs in the MASTER address space on those other systems.

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 z/OS or z/VM.

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 (z/OS 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 data set name table 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 112.

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 318 and “Emergency data set name tables” on page 49.

Enabling sysplex communication

A flag in the RACF data set name table 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.

Guideline: Use a common data set name table if you can. 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 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 40.

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* 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 the data set name table, 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
is the number of I/O buffers defined in the data set name table. (For backup data sets use 20% of the value specified for the primary).

4K
is the size of the buffer

b/10
is 10% of the total number of buffers per system, to allow different data reference patterns

n
is the number of CPCs in a sysplex

C
is 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 that 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.

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 318 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.

Guideline: Define the user ID assigned to a started procedure to be a protected user ID, so that the user ID cannot be revoked by incorrect password or password phrase attempts or used to enter the system in ways that require a password or password phrase.

To define a user ID as protected, assign it the NOPASSWORD, NOPHRASE, 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.

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 authorization requests done for the procedure are considered successful, without actually performing any checking. This includes bypassing the checks for security classification on users and data. Additionally, the following processing is affected.

- For RACROUTE REQUEST=AUTH, RACF:
  - Does not call any exit routines
  - Does not generate any SMF records
  - Does not update any statistics
- For RACROUTE REQUEST=FASTAUTH, RACF
  - Calls exit routines as usual
  - Does not generate any SMF records
  - Does not update any statistics
- For RACROUTE REQUEST=DEFINE,
  - The checking done for the CHKAUTH operand is bypassed
  - All other RACF processing occurs as usual

RACF allows you to specify that a started procedure is *trusted*; this means that most authorization requests done for the procedure are considered successful, without actually performing any checking. This includes bypassing the checks for security classification on users and data. Additionally, the following processing is affected.

- For RACROUTE REQUEST=AUTH, RACF:
  - Does not call any exit routines

  - Generates 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
- For RACROUTE REQUEST=FASTAUTH, RACF
  - Calls exit routines as usual
  - Generates SMF records based on the UAUDIT setting in the user ID profile
  - Does not update any statistics
- For RACROUTE REQUEST=DEFINE,
  - The checking done for the CHKAUTH operand is bypassed
  - 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.

Guideline: Assign the TRUSTED attribute when one of the following conditions applies:

- The started procedure or address space creates or accesses a wide variety of unpredictably named data sets within your installation.
- Insufficient authority to an accessed resource might risk an unsuccessful IPL or other system problem.

For a list of required and optional candidates for the TRUSTED attribute, see Assigning the RACF TRUSTED attribute in *z/OS MVS Initialization and Tuning Reference*.

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.

Note:

1. If ENTITY=(...,CSA) or ENTITY=(...,PRIVATE) is coded on the RACROUTE REQUEST=AUTH macro instruction, RACF ignores the privileged and trusted attributes and performs normal authorization processing.
2. If the requested class is defined, active, and appropriately RACLISTed (if required), then except as mentioned in Note “1” on page 107, a started procedure that has the privileged or trusted attribute accepts any checking done by RACROUTE REQUEST=AUTH and RACROUTE REQUEST=FASTAUTH, including security classification 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 112.)
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 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 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 and 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 various 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 name or 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 about these commands, see *z/OS Security Server RACF Command Language Reference*.

You should 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 overrides all the entries in ICHRIN03.

For additional information about the STARTED class, see *z/OS Security Server RACF Security Administrator's Guide*. For information about 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 110.

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 current versions of RACF . 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 110.
  - 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 6 on page 110. 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 6. 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, you should include 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 entry 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 documents 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'
	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.

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, PHRASE, NEWPHRASE, ICTX, ICRX, and IDID keywords. The programs must be reentrant and fetched from an APF-authorized library.

Guidelines:

- Because incorrect use of ICHAUTAB can cause system integrity problems, do not use it; instead 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 356.
- 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 system 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*, available at CICS Transaction Server for z/OS (www.ibm.com/support/knowledgecenter/SSGMGV/welcome.html).

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.

TXSeries

IBM TXSeries® for Multiplatforms includes the CICS application servers for AIX®, Solaris, HP-UX and Windows systems. TXSeries can use information from the RACF database to define user information on distributed platforms. For more information about TXSeries, including the TXSeries library, see TXSeries for Multiplatforms (www.ibm.com/software/products/txseries).

DFSMS considerations

RACF allows DFSMS to establish defaults for the 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.

In a data set profile, 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.

In a data set profile, the DFP field, DATAKEY, contains the CKDS key label that SMS will associate with a data set protected by the profile at time of allocation.

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 318.

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 Db2 RACF access control 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 *RACF Access Control Module Guide* for your version of Db2. You can find Db2 information at IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

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 122.

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
- ICKDSF
- IEBCOMPR
- IEBCOPY
- IEBDG
- IEBEDIT
- IEBGENER
- IEBISAM
- IEBPTPCH
- IEBUPDTE
- IEHLIST
- IEHMOVE

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).

Note:

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 249 and the description of the ICHNCONV macro in *z/OS Security Server RACF Macros and Interfaces*.

2. See Chapter 6, "RACF database utilities," on page 203 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 do the following:

- 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 noted 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*
- *Device Support Facilities (ICKDSF) 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 (a z/OS system with RACF enabled 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 (a z/OS system with RACF enabled 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. 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 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 z/OS, 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.

Note:

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 enabled:

- 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

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 enabled 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 DFSMS) 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:

- IRXREXX1 for IRXPARMS
- IRXREXX2 for IRXTSPRM
- IRXREXX3 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*.

Initializing RACF verification of signed programs (IRRVERLD)

The IRRVERLD program initializes RACF verification of signed programs by loading and verifying the RACF program verification module (IRRPVERS). IRRPVERS is a signed program that performs verification of signed programs, and must be loaded and verified before RACF is able to verify other signed programs. Before you run IRRVERLD, the security administrator must configure RACF to verify signed programs, and must define IRRPVERS as a signed program that must be verified when it is loaded. See Enabling RACF to verify signed programs in *z/OS Security Server RACF Security Administrator's Guide* for more information about implementing program signature verification.

Important: Run the IRRVERLD program on each system which shares this RACF database.

IRRVERLD has no parameters and is run using JCL. For example:

```
//IRRVERLD    JOB
//IRRVERLD    EXEC   PGM=IRRVERLD
```

If IRRVERLD does not successfully complete, work with your RACF security administrator to resolve error messages and rerun IRRVERLD. After IRRVERLD successfully completes, the IRRPVERS module is active and RACF is enabled to verify signed programs. The IRRPVERS module remains active until the next IPL. At IPL time, RACF initialization automatically reloads and verifies IRRPVERS unless the RACF configuration for program signing is disabled between IPLs. You can rerun IRRVERLD at any time to check the status of the IRRPVERS module.

IRRVERLD return codes

The IRRVERLD program sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	IRRVERLD completed successfully. The IRRPVERS module was loaded and verified. RACF program verification is active.

Hex	(Decimal)	Meaning
4	(4)	The IRRPVERS module was previously loaded and verified. RACF program verification is already active.
8	(8)	An error occurred during the load and verification of IRRPVERS. RACF program verification is not active.
10	(16)	A severe error occurred.

Chapter 5. RACF remote sharing facility (RRSF)

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 and TCP/IP 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 154.

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 191 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 "Overview of the RRSF function" on page 127) and allows the associated user IDs to synchronize their passwords and password phrases (see "Overview of the RRSF function" on page 127). 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 191 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

RRSF provides the following functions:

© Copyright IBM Corp. 1994, 2020

- 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. A user directs a command by using the AT keyword on the command to specify the RRSF node and user ID the command is to be directed to. The command runs asynchronously in the RACF subsystem address space, and the output is returned to the issuing user's RRSFLIST data set.

  Before a user can direct a command to run under another user ID, a user ID association must be established between the two user IDs. Profiles in the RRSFDATA class control to which nodes command direction is allowed, and which users can direct commands.

- Password synchronization

  If password synchronization is enabled between two user IDs, when the password or password phrase is changed for one of the user IDs, RACF automatically changes the password or password phrase for the other.

  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. The SET command activates and deactivates password synchronization.

- 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. Automatic direction does not require user ID associations. Instead, automatic direction assumes that if the same user ID exists on two different nodes, those user IDs belong to the same person. RACF provides the following types of automatic direction:

  – Automatic command direction. Profiles in the RRSFDATA class control which commands are automatically directed, and to which nodes.
  – Automatic password direction. Profiles in the RRSFDATA class control for which users password and password phrase changes are automatically directed, and to which nodes.
  – Automatic direction of application updates. Profiles in the RRSFDATA class control which application updates are automatically directed to which nodes.

  The SET command activates and deactivates automatic direction.

For a more detailed description of these functions, see *z/OS Security Server RACF Security Administrator's Guide*.

The RRSF network

An RRSF network is a collection of RRSF nodes in a network using APPC/MVS or TCP/IP as the network transport mechanism. Figure 7 on page 129 illustrates an RRSF network.

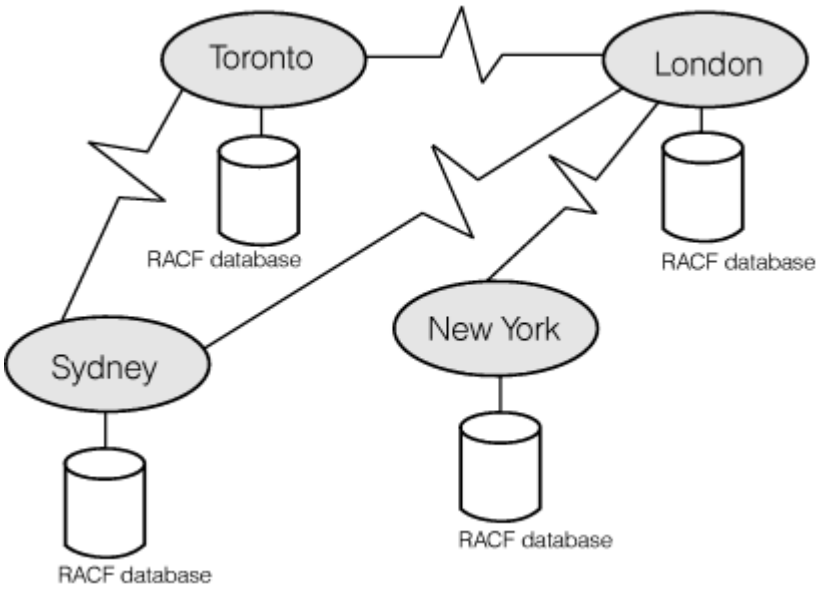


Figure 7. An RRSF network

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 using the TARGET command" on page 159 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:

- The RACF component of the z/OS Security Server is enabled.
- 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 7 on page 129, 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 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 8 on page 130 shows an RRSF network containing a single-system node and a multisystem node.

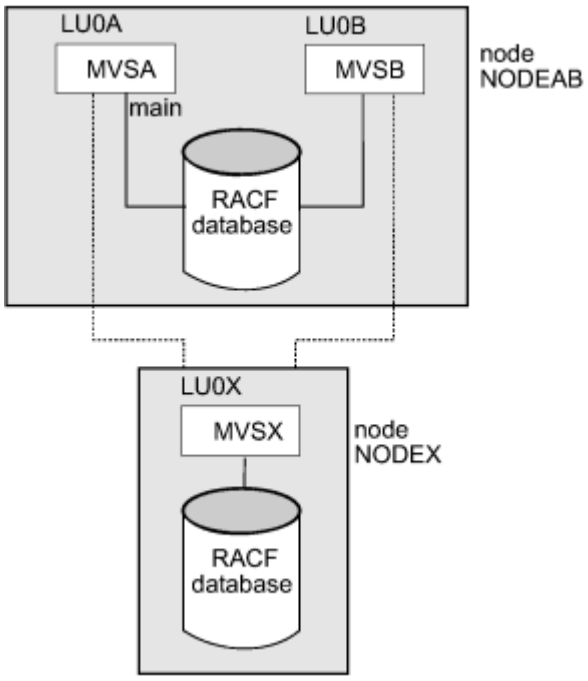


Figure 8. An RRSF network containing a single-system node and a multisystem node

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 and password phrase 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 by way of 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 and password phrase changes when automatic password direction is used. These are returned to the system on which the password or password phrase 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 9 on page 131 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.

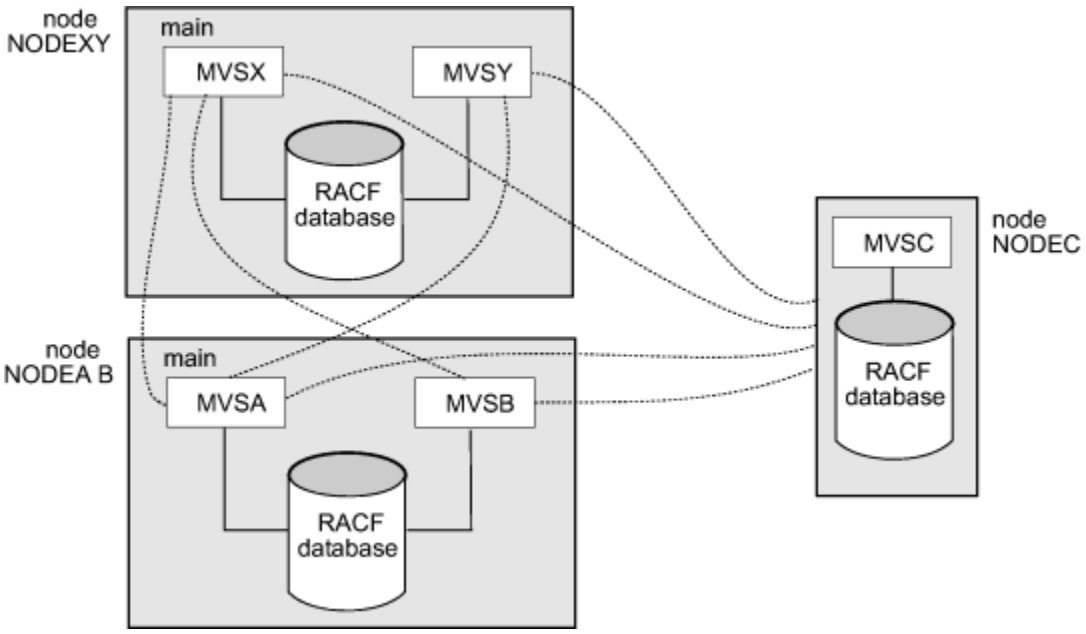


Figure 9. An RRSF network with two multisystem nodes and one single-system node

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 of operation

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 and password phrase 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.

If you define a node to communicate with another node using the APPC/MVS protocol, both the local mode functions and the remote mode functions must initialize successfully before the connection can enter the operative active state. An error in initializing the local mode functions (for example, a VSAM error on the local nodes's workspace data sets) prevents remote connections from being established. An error in initializing the remote mode functions (for example, an APPC server initialization failure) prevents local mode functions from being performed.

If you define a node to communicate with another node using the TCP/IP protocol, an error in initializing the local mode functions (for example, a VSAM error on the local nodes's workspace data sets) prevents

remote connections from being established. However, unlike the APPC/MVS protocol, an error in initializing the remote mode functions for TCP/IP does not prevent the local mode functions from being performed.

If a node communicates with other nodes using both TCP/IP and APPC/MVS, an error initializing the APPC remote functions prevents the TCP/IP remote functions from being used. However, an error starting the TCP/IP remote functions does not prevent the APPC remote functions from being used.

RACF creates a *listener* process on the local node for each protocol that the node uses. The listener process listens for incoming connection requests from remote nodes. The local node cannot establish new remote connections for a protocol unless its listener for the protocol is active. The listener for a protocol can be in one of the following states:

active
The listener has been established and is listening for connection requests from remote nodes. New connections can be established with remote nodes.

inactive
The listener is not currently available. This state can occur when the local node has been made dormant, or, for the TCP/IP listener, if there is a problem with z/OS UNIX. Remote connections that are already active remain active and continue to communicate, but new connections cannot be established. To make the listener active, issue a TARGET OPERATIVE command or a RESTART CONNECTION command for the local node.

initializing
The listener is attempting to start, but has not been able to start successfully. The listener will retry periodically until it starts successfully or the local node is made dormant. If the listener cannot start within a time period of approximately 30 minutes, it stops retrying and goes into inactive state. The initializing state can occur when there is a problem with TCP/IP, or if a host name was specified for the local node and the domain name server (DNS) is not responding for host name resolution, or if an incorrect host name or IP address was specified. Remote connections that are already active remain active and continue to communicate, but new connections cannot be established.

The TARGET LIST command for the local node displays the status of the listener processes for the local node.

Connections between nodes

Two RRSF nodes are said to be *logically connected* when they are configured to communicate by way of APPC/MVS or TCP/IP, 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, as documented in “Connection states” on page 133.

Operative connections

When a node's connection to a remote node is fully *operative* (in the operative active state, see “Connection states” on page 133), outgoing requests from the node are sent immediately to the remote node. A connection goes through several operative states, described in “Connection states” 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 172 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 136 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 172 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 13 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 163 for more information about TARGET LIST. For further information about connection states and transitions between them, see *z/OS Security Server RACF Diagnosis Guide*.

Table 13. 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. This state can occur while the local node is attempting to restore a successful connection that was interrupted.
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. The information can include RACF information and the digital certificates exchanged during the TLS handshake performed by TCP/IP. 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.

Table 13. Connection states between nodes (continued)

Name	Abbreviation	Description
operative in error	O-E	A non-retryable error has occurred related to workspace data sets (for example, a volume has gone offline). Console or syslog messages have been issued with more information.
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 APPC connection between the local and remote nodes has not been defined on the remote node. This state can occur if the local node cannot successfully connect with a node after a reasonable amount of time.
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.
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. The TARGET LIST command output shows this state as ???.

Network protocols

RACF supports the following network protocols for an RRSF network:

- TCP/IP

- APPC/MVS and VTAM

You can use one or both of these protocols in an RRSF network. IPv4 and IPv6 are supported for TCP/IP. See *z/OS Communications Server: IPv6 Network and Appl Design Guide* for more information about the IPv6 protocol in z/OS Communication Server.

Using TCP/IP in an RRSF network

z/OS Communications Server provides the TCP/IP networking protocol on z/OS. It also provides Application Transparent Transport Layer Security (AT-TLS), which allows client and server applications to communicate safely using TCP/IP. RACF uses AT-TLS to provide authentication between RRSF nodes and to provide encryption of RRSF traffic. RACF does not allow RRSF nodes to connect unless the connection is protected by an AT-TLS rule enforcing client authentication.

A sample AT-TLS policy for RRSF is included in the Configuration Assistant for z/OS Communications Server, which is available as an optional plug-in for IBM z/OS Management Facility (z/OSMF). (For information about z/OSMF, visit the z/OS Management Facility home page (www.ibm.com/systems/z/os/zos/features/zosmf).) Also, RACF ships sample policy statements in the IRRSRRSF member of SYS1.SAMPLIB. (Note that this policy specifies a different key ring name than the default specified in the Configuration Assistant). You can edit these statements into your existing policy.

A default port number of 18136 has been reserved with the Internet Assigned Numbers Authority (IANA) for the TCP/IP listener socket. The TARGET command defaults to this value for the port number, and this is the port number specified in the sample AT-TLS policy for RRSF provided by z/OS Communications Server.

For information about setting up your system to use TCP/IP in an RRSF network, see “Setting up your system to use TCP/IP” on page 153.

Using APPC/MVS and VTAM in an RRSF network

APPC/MVS is a communications vehicle for sending and receiving messages from one RRSF node to another. If you use it, you must configure VTAM for APPC/MVS and implement APPC/MVS before you can use RRSF in remote mode. For information about setting up your system to use APPC/MVS and VTAM in an RRSF network, see “Setting up your system to use APPC/MVS and VTAM ” on page 149.

Using mixed protocols

An RRSF network can use both APPC/MVS and TCP/IP at the same time. A given node can communicate with some nodes using APPC/MVS, and with some using TCP/IP. To implement a node that communicates using multiple protocols, you must specify multiple protocols in its local node definition. To do this you need to issue multiple TARGET LOCAL commands, because you can specify only one protocol on a TARGET command. Each set of protocol information is referred to as a *protocol instance*. RACF establishes a listener process on the node for each protocol instance. A connection to a remote node uses only one protocol at a time, and you specify only one protocol for a remote node. If you specify a second protocol for a remote node, RACF converts that connection from the first protocol to the second. (For information about the conversion process, see “Changing the protocol for a connection” on page 181.)

Encryption and masking of data

Data in an RRSF network is protected by the following means:

- RACF masks the data on the RRSF message queues and during transmission.
- The network protocol encrypts the data during transmission.

Masking of data

RACF masks the data portion of RRSF message packets. The data is masked while the message packets are on the RRSF message queues, saved in the workspace data sets, 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 encryption by the network protocol provides 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.

Encryption of data

APPC/MVS encrypts RRSF data during transmission using DES with an effective key length of 56 bits.

TCP/IP uses AT-TLS to encrypt RRSF data. AT-TLS provides a number of cipher suites, most of which are stronger than the DES used by APPC/MVS. When you set up the AT-TLS policy, you can specify a number of cipher suites within a rule in order of preference, and AT-TLS selects the first one requested by both of the communicating nodes. For information about the cipher suites that AT-TLS supports, see the description of the TTLSCipherParms statement in *z/OS Communications Server: IP Configuration Reference*. After successfully connecting, RACF issues a message indicating the cipher in effect. RACF does not enforce a minimum encryption level, and allows a connection with no encryption. (You might want to specify no encryption for AT-TLS if you have specified encryption at the link layer, or if your nodes are connected across LPARs on the same physical system and you are willing to trade off the low level of risk for improved performance.) The sample AT-TLS rules for RRSF specify 256-bit AES encryption.

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 140 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 for the local node is used to temporarily hold requests that are sent to the local node from itself. The INMSG data set for a remote node is used to temporarily hold requests that are sent to the local node from the 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 conventions for the workspace data sets depend on whether the node is local or remote, and, for remote nodes, on which protocol (APPC or TCP/IP) is being used.

Workspace data sets for a local node

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

Workspace data sets for a remote node using APPC

The naming convention for the workspace data sets for remote connections that use the APPC protocol 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.

Workspace data sets for a remote node using TCP/IP

The naming convention for the workspace data sets for remote connections that use the TCP/IP protocol is:

```
prefix.local_node_qualifier.sysname_or_wdsqual_or_nodename.ds_identity
```

where:

prefix
Is a value you specify with the PREFIX keyword on the TARGET command

local_node_qualifier
Is the middle qualifier for the local node. This defaults to the system name contained within the CVTSNAME field, but can be overridden with the WDSQUAL keyword on the TARGET LOCAL command.

sysname_or_wdsqual_or_nodename
If the remote node is a single-system node, the node name specified in the NODE keyword on the TARGET command is used by default. If the remote node is a multisystem node, the system name specified in the SYSNAME keyword on the TARGET command is used by default. If a WDSQUAL value is specified on the TARGET command, it overrides the defaults.

ds_identity
Is either INMSG or OUTMSG

Examples of workspace data set names

Table 14 on page 138 shows the workspace data sets that are created for the RRSF network shown in Figure 30 on page 197, for both the APPC and TCP/IP protocols.

Table 14. Example of workspace data set names

If the protocol is APPC ...	**If the protocol is TCP/IP ...**
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 MVSA: SYS1.MVSA.INMSG SYS1.MVSA.OUTMSG SYS1.MVSA.NODEX.INMSG SYS1.MVSA.NODEX.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 MVSB: SYS1.MVSB.INMSG SYS1.MVSB.OUTMSG SYS1.MVSB.NODEX.INMSG SYS1.MVSB.NODEX.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	The following workspace data sets are created on system MVSX: SYS1.MVSX.INMSG SYS1.MVSX.OUTMSG SYS1.MVSX.MVSA.INMSG SYS1.MVSX.MVSA.OUTMSG SYS1.MVSX.MVSB.INMSG SYS1.MVSX.MVSB.OUTMSG

Defining the workspace 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 using the TARGET command” on page 159 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.

Make sure that the RACF subsystem user ID has the authority to create and access the workspace data sets. The suggested way to do this is to define the subsystem as trusted or privileged. For more information, see “Assigning a user ID to the RACF subsystem” on page 80 .

There are two methods you can use to allocate the workspace data sets:

- Let RACF allocate the VSAM data sets for you.

  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. RACF uses this information to allocate the data sets. For more information, see “Defining RRSF nodes to RACF using the TARGET command” on page 159.
- Preallocate the VSAM data sets yourself.

  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. Note that RACF might delete data sets that you have preallocated. See "Deleting the workspace data sets" on page 140.

Whenever a TARGET command is issued to make a node operative or dormant, RACF uses the information on the WORKSPACE and PREFIX keywords to derive the names of the workspace data sets. If data sets with those names do not exist, RACF allocates new data sets. If the data sets already exist, (because either you or RACF has already allocated them), RACF uses the existing data sets.

If a node is using a set of workspace data sets, and you edit the TARGET commands in the RACF parameter library so that RACF derives different data set names on the next IPL, RACF allocates workspace data sets with the new names. The old data sets are not deleted, but any requests queued in them are ignored. If you later change the TARGET commands to specify the old data sets again, the old requests are still there and RACF processes them.

If you issue a TARGET command with new values for the WORKSPACE and PREFIX 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.

Modifying the attributes of workspace data sets

The attributes of the workspace data sets can be modified for any operative active remote or local node through the use of the NEWPREFIX and NEWWORKSPACE operands of the TARGET command. There are a number of rules and restrictions which are documented in the command language reference for these keywords.

When NEWPREFIX and NEWWORKSPACE are specified for an RRSF node on the TARGET command, a second set of workspace data sets is allocated for new work, while existing records in the old workspace data sets are processed. When the old workspace data sets are empty, they are deleted.

Because both old and new workspace data sets are open simultaneously, the new workspace data sets must have different names. The NEWPREFIX operand of the TARGET commands may not specify the same prefix that is currently in use for that node.

The conversion from old workspace data sets to new workspace data sets begins immediately after the TARGET command which specifies the NEWPREFIX and NEWWORKSPACE operands is issued.

IBM recommends that workspace data set changes are made during periods of low RRSF activity when the existing workspace data sets are as close to empty as possible.

The PREFIX, WDSQUAL and all sub-operands of the WORKSPACE operand can be modified using NEWPREFIX and NEWWORKSPACE. There is no NEWWDSQUAL keyword, but WDSQUAL can be specified with NEWPREFIX and NEWWORKSPACE if a different value is needed.

Suboperand values of the NEWWORKSPACE keyword are copied from the existing WORKSPACE values for the node being modified. Suboperands can be specified for NEWWORKSPACE to override the current WORKSPACE values.

Changing the attributes of the workspace data sets without changing the name is only possible using two TARGET commands. The first TARGET command specifies a temporary prefix to allow the old workspace data sets to be emptied and deleted. The second TARGET command re-specifies the original, desired workspace data set prefix. Ensure that RACF profiles exist to protect the temporary workspace data sets during the conversion process.

If pervasive encryption is enabled in the RACF profile which protects the new workspace data sets, the new workspace data sets will be encrypted when they are allocated. Use the two TARGET command mechanism described above to encrypt the current workspace data sets using pervasive encryption.

The NEWPREFIX and NEWWORKSPACE operands of the TARGET command cannot be used to recover from a full data set condition. They can only be used in an error-free environment.

Note that network communications are automatically dropped and re-established during the workspace data set conversion procedure.

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 331.

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 10 on page 141 illustrates the journey of a directed RACF command through an RRSF network. The network contains two RRSF nodes, NODEA and NODEB, that use APPC to communicate. 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.

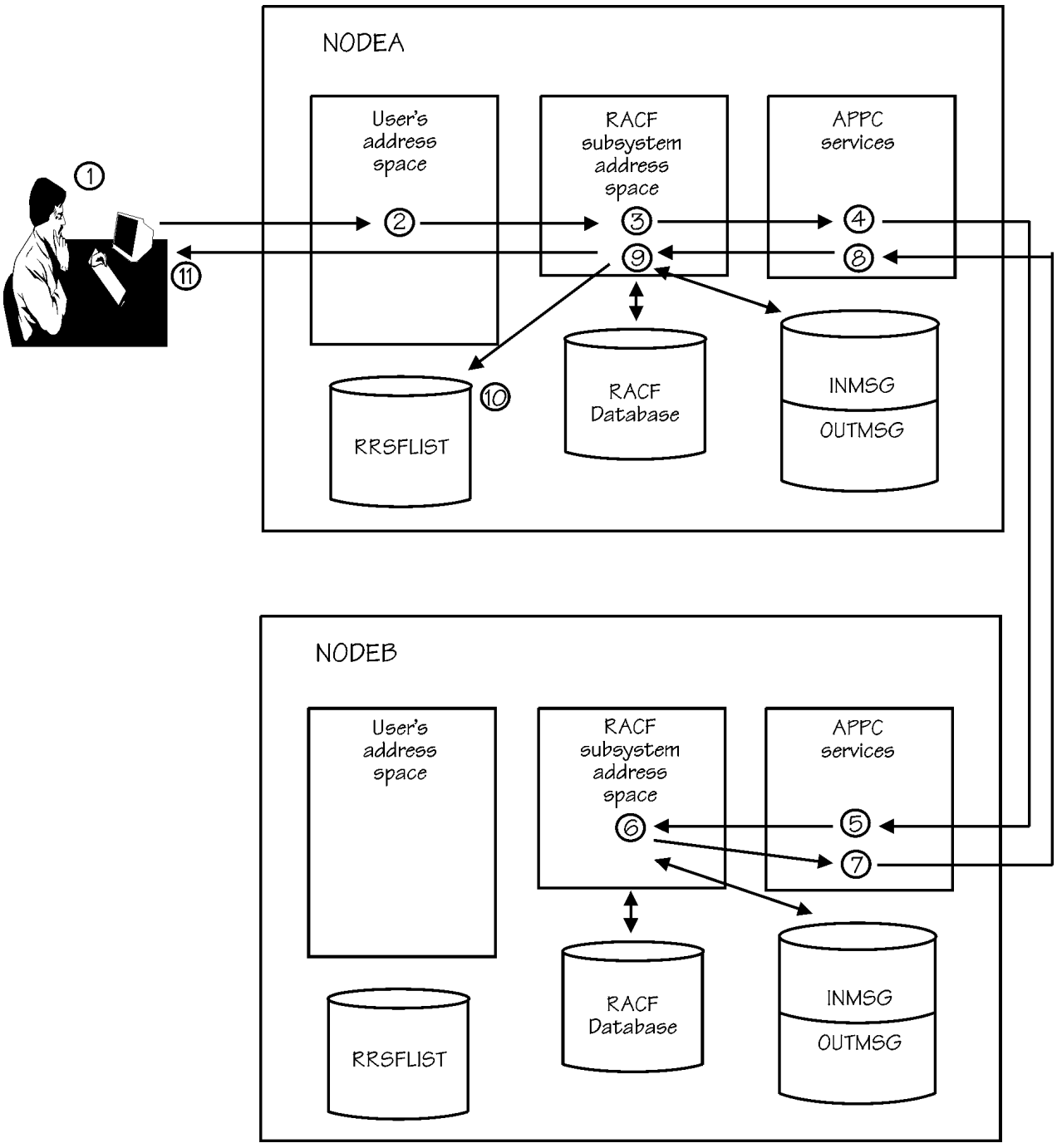


Figure 10. A directed command traveling through an RRSF network that uses APPC

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 by way of 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 by way of 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 “2.b” on page 141 would have been replaced by a check that user ID PAT on NODEA had SPECIAL authority.
- Step “6.e” on page 142 would have additionally checked whether user ID PATL on NODEB had SPECIAL authority.

Defining an RRSF environment

Table 15 on page 143 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, some might be performed by a network administrator, and some might be performed by all of these team members working together.

Table 15. 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 for which you are using APPC as the protocol. • Find out the TCP/IP hostname or the IP address for each node for which you are using TCP/IP as the protocol. • 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 191
Ensure that all systems to be in the environment have enabled the RACF component of the z/OS Security Server.	"System prerequisites" on page 145
Evaluate whether you require cryptographic teleprocessing support, and implement this if required.	"Encryption and masking of data" on page 135
Ensure that the RACF template versions are compatible on all systems.	"RACF template version considerations" on page 145
Ensure that the RACF dynamic parse versions are compatible on all systems.	"RACF dynamic parse version considerations" on page 146
Ensure that the SETROPTS option settings are compatible on all systems.	"SETROPTS options considerations" on page 146
Ensure that installation exits are compatible on all systems.	"Installation exit considerations" on page 147
Ensure that password authentication algorithms are sufficient on all systems.	"Installation exit considerations" on page 147

Table 15. Defining an RRSF environment—summary of tasks. Tasks that are not numbered can generally be done in any order. *(continued)*

Task	For more information, refer to ...
If you are using APPC/MVS, 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.	“Setting up your system to use APPC/MVS and VTAM ” on page 149
If you are using TCP/IP, set up TCP/IP and AT-TLS • Protect the RRSF listener port • Set up AT-TLS • Prevent RACF from attempting remote communications before the AT-TLS policy is available • Ensure that the RACF subsystem address space can access the TCP/IP stack • Allow the subsystem address space to use z/OS UNIX socket APIs	“Setting up your system to use TCP/IP” on page 153
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 148
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 147
If you are planning to have RACF maintain synchronization of any profiles between databases, synchronize those profiles.	“Synchronizing database profiles” on page 147
Create or modify the JCL to activate the RACF subsystem. Make sure the user ID for the RACF subsystem can access the RRSF resources. If you plan to use TCP/IP, make sure that the user ID for the RACF subsystem has a UID, and that its default group has a GID.	“RACF subsystem address space considerations” on page 149
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 154

Table 15. Defining an RRSF environment—summary of tasks. Tasks that are not numbered can generally be done in any order. *(continued)*	
Task	**For more information, refer to ...**
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 184
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 using the TARGET command" on page 159
Customize the RRSF environment by defining RRSFDATA profiles.	"Customizing a remote sharing environment" on page 191
Activate the RRSFDATA class on each RRSF node.	"Customizing a remote sharing environment" on page 191
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 149

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 functions on each node in the network:

- The RACF component of the z/OS Security Server is enabled

and for remote communications:

- APPC (part of MVS) and VTAM or TCP/IP and AT-TLS (provided by the z/OS Communications Server)

Nodes configured in local mode do not require APPC, VTAM, TCP/IP, or AT-TLS.

To configure two nodes to communicate using the TCP/IP protocol, both nodes must be running z/OS V1R13.

When you use the TARGET command to establish communications between two RRSF nodes, RACF expects that both nodes are running RACF, and that both nodes are running either APPC or TCP/IP and AT-TLS.

Before you can define a multisystem node in an RRSF network, *each* system in the network must have the RACF component of the Security Server enabled.

To have application updates directed to or from a node, the node must have the RACF component of the Security Server enabled.

For TCP/IP connections, you must implement an RRSF trust policy based on digital certificates. If you store the private keys for any of these digital certificates in the ICSF PKA key data set (PKDS), you must ensure that ICSF starts during IPL before the Policy Agent, or RRSF connections will fail. For information about setting up the RRSF trust policy, see *z/OS Security Server RACF Security Administrator's Guide*.

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 209.

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 and IRRDPI00" on page 67 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*)

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 about the SETROPTS command, see *z/OS Security Server RACF Command Language Reference*.

Password rules

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.

Mixed case passwords

Problems can occur when RRSF nodes differ in whether they allow mixed case passwords. The MIXEDCASE and NOMIXEDCASE parameters on the PASSWORD option of the SETROPTS command determine whether the system allows mixed case passwords.

Rules: The following rules apply to the case of a password when it is propagated:

- When passwords are propagated to a system with NOMIXEDCASE in effect, the result on the target system is a password in upper case.
- When passwords are propagated to a system with MIXEDCASE in effect from a system with NOMIXEDCASE in effect, the results differ depending on the type of propagation used:
  1. With password synchronization or automatic password direction, if an application (for example, TSO LOGON) sets the new password entered by the user to upper case before passing it to RACF, the resulting password is in upper case on both systems. If, however, an application sets a lower case password directly via ICHEINTY, the password is in lower case on both systems.
  2. With command direction or automatic command direction, the command is sent to the other system as entered. So if the user types a lower case password, the result is an upper case password on the NOMIXEDCASE system, and a lower case password on the MIXEDCASE system.

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.

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. You can get this tool and others from the RACF home page.

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 it is a good practice to 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 285.

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 need to remove that code. You must 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 by way of 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 77 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 184 for information on the RACF parameter library. See "The RACF PROC" on page 82 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

Guideline: Define this user ID to be trusted or privileged. See "Assigning a user ID to the RACF subsystem" on page 80.

If your RRSF network uses the TCP/IP protocol, the RACF subsystem must be able to use z/OS UNIX socket APIs. To allow the subsystem to use these APIs, you must ensure that user ID for the RACF subsystem is assigned a UID, and that its default group is assigned a GID.

Setting up your system to use APPC/MVS and VTAM

This document assumes that you have a basic understanding of VTAM and APPC/MVS. For information about 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 that uses the APPC protocol, 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.

Guideline: Protect the information flowing between RRSF nodes by specifying VERIFY=REQUIRED on the APPC LU (ACB) definitions in the SYS1.VTAMLST library concatenation.

If you specify VERIFY=REQUIRED, then to obtain 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 configured 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

You should 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 151), 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 “Setting up your system to use APPC/MVS and VTAM ” on page 149 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
```

Guideline: Define this profile with a UACC of NONE:

```
RDEFINE FACILITY APPCMVS.DBTOKEN UACC(NONE)
```

Then use PERMIT commands to give user IDs and groups 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.

Setting up your system to use TCP/IP

The Transmission Control Protocol (TCP) and the Internet Protocol (IP) is a protocol suite that allows communications in a network. IPv4 and IPv6 are supported for TCP/IP. See *z/OS Communications Server: IPv6 Network and Appl Design Guide* for more information about the IPv6 protocol in z/OS Communication Server. On z/OS, the Communications Server provides support for TCP/IP. For information about configuring TCP/IP, see:

- *z/OS Communications Server: IP Configuration Guide*
- *z/OS Communications Server: IP Configuration Reference*

You need to perform the following tasks to set up your RRSF network to use TCP/IP:

- Protect the RRSF listener port. For information, see "Protecting the RRSF listener port" on page 153.
- Set up AT/TLS. For information, see "Setting up AT-TLS" on page 153.
- Prevent RACF from trying to establish connections before the AT-TLS policy is available. For information, see "Stack initialization protection" on page 154.
- Give the RACF subsystem address space access to the TCP/IP stack. For information, see "Stack access control" on page 154.
- Allow the RACF subsystem address space to use z/OS UNIX socket APIs. For information, see "Allowing the subsystem address space to use z/OS UNIX socket APIs" on page 154.

Protecting the RRSF listener port

Protect the RRSF listener port so that only the RACF subsystem address space has access to it. A default port number of 18136 has been reserved with the Internet Assigned Numbers Authority (IANA) for the RRSF listener port. Define the RRSF port in the TCP/IP profile, and assign it a name using the SAF keyword. For information about protecting ports, see the section on controlling access to particular ports in *z/OS Communications Server: IP Configuration Guide*. For information about the PORT command, see *z/OS Communications Server: IP Configuration Reference*. For example, to assign the name RRSF to the listener port, you might create the following entry:

```
PORT 18136 TCP * SAF RRSF ; Listener port for RACF Remote Sharing
```

To protect access to the port, the security administrator creates a profile in the SERVAUTH class. For more information, see Establishing RACF security for RRSF TCP/IP connections in *z/OS Security Server RACF Security Administrator's Guide*.

Setting up AT-TLS

RACF relies on AT-TLS to authenticate the RRSF nodes, and refuses to accept an RRSF connection unless AT-TLS has performed client authentication. Therefore, you must enable and configure AT-TLS. For information about how to do this, see the chapter on Application Transparent Transport Layer Security data protection in *z/OS Communications Server: IP Configuration Guide*. The security administrator must implement a trust policy based on digital certificates for AT-TLS. For more information, see Implementing an RRSF trust policy in *z/OS Security Server RACF Security Administrator's Guide*. If you store the private keys for any of these digital certificates in the ICSF PKDS, you must ensure that ICSF starts during IPL before the Policy Agent, or RRSF connections fails.

z/OS Communications Server provides a sample AT-TLS policy in its IBM Configuration Assistant for z/OS Communications Server. Also, RACF ships sample policy statements in the IRRSRRSF member of SYS1.SAMPLIB, that you can edit into your existing policy.

The sample AT-TLS policy that z/OS Communications Server provides is shipped disabled, and you must enable it, and install it into Policy Agent. Some important features of the policy are:

- It consists of two rules: one to describe RRSF as the "server" role, and one for the "client" role.
- The server role specifies a client authentication level of "Required". You can specify "SAFCheck" instead, for increased security, but you only need to do this if you do not have full control over your signing certificate. (For more information about specifying "SAFCheck", see *z/OS Security Server RACF Security Administrator's Guide*.) Do not specify "Full". For a description of the different levels of client

authentication, see the section on TLS/SSL security in *z/OS Communications Server: IP Configuration Guide*.

- It specifies the AES 256 cipher TLS_RSA_WITH_AES_256_CBC_SHA, which means 256-bit AES encryption with SHA-1 message authentication and RSA key exchange. RACF does not enforce a minimum encryption level.
- It specifies the default listening port number 18136 for the server. The policy matches a client connection from any ephemeral port number.
- It specifies a default key ring name of `tlsKeyring`. There is no dependency on this name within RRSF, so you can specify a different name that is not used by another application. (The sample policy that RACF provides in the IRRSRRSF member of SYS1.SAMPLIB specifies `IRR.RRSF.KEYRING`.)
- It specifies only the level TLS V1.1 of the TLS protocol.
- It specifies no application control.

You can use the NETSTAT command provided by z/OS Communications Server to display detailed information about the AT-TLS policy covering an RRSF connection. For more information, see *z/OS Security Server RACF Diagnosis Guide* and *z/OS Communications Server: IP System Administrator's Commands*.

Stack initialization protection

During a system IPL, if the RACF subsystem address space initializes before the TCP/IP and Policy Agent address spaces, it is possible that RACF might try to establish a listener socket and then attempt to establish remote connections before the AT-TLS policy is available. If this happens, the connections are rejected and you will need to establish them manually after the IPL completes. The resource EZB.INITSTACK.*sysname*.*tcpname* in the SERVAUTH class controls the ability of applications to open a socket before the AT-TLS policy is loaded onto the TCP/IP stack. To prevent RACF from trying to establish connections before the AT-TLS policy is available, do not give the RACF subsystem address space user ID access to this resource if a profile is defined for it.

When the RACF subsystem address space user ID does not have access to the INITSTACK profile, and it attempts to establish a listener socket, an ICH408I message is issued to the console. RACF retries the attempt to establish a listener socket until the AT-TLS policy is available and the attempt succeeds. Depending on the length of time it takes for Policy Agent to initialize, a number of the ICH408I messages might be issued. You can ignore them.

For more information, see the section on TCP/IP stack initialization access control in *z/OS Communications Server: IP Configuration Guide*.

Stack access control

Stack access control provides a way to allow or disallow users or groups of users to access a TCP/IP stack. The TCP/IP stack to be protected is represented by the resource EZB.STACKACCESS.*sysname*.*tcpname* in the SERVAUTH class. If you have a RACF profile protecting this resource, the RACF subsystem address space user ID must have access to it.

For more information, see the section on stack access control in *z/OS Communications Server: IP Configuration Guide*.

Allowing the subsystem address space to use z/OS UNIX socket APIs

The RACF subsystem must be able to use z/OS UNIX socket APIs to communicate via TCP/IP in an RRSF network. To allow the subsystem to use these APIs, you must ensure that the user ID for the RACF subsystem is assigned a z/OS UNIX UID, and that its default group is assigned a GID. For information about assigning UIDs and GIDs, see *z/OS Security Server RACF Security Administrator's Guide*.

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 that uses APPC/MVS" on page 195 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 perform the following actions related to RRSF:

- List the attributes of the local node. See "Listing information about RRSF functions on the local node" on page 155.
- Specify trace parameters. See "Tracing APPC, IMAGE, and RRSF events" on page 156.
- 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 184.
- 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.
- Prepare a multisystem node for dynamic MAIN system switches. See "Allocating workspace files for remote non-MAIN connections in preparation for dynamic main switches" on page 157 in preparation for dynamic MAIN system switches.

The SET command can be issued as a RACF operator command, or from the RACF parameter library. For 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 information about RRSF functions on 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 11 on page 156 illustrates the type of information displayed.

```
         IRRH005I (@) RAC1 SUBSYSTEM INFORMATION:
            TRACE OPTIONS                   - NOIMAGE
                                            - NOAPPC
                                            - NOSYSTEMSSL
                                            - NORRSF
                                            - NORACROUTE
                                            - NOCALLABLE
                                            - NOPDCALLABLE
                                            - NODATABASE
                                            - NOGENERICANCHOR
                                            - NOASID
                                            - NOJOBNAME
                                            - NOCLASS
                                            - NOUSERID
            SUBSYSTEM USERID                - RRSFUSER
            JESNODE (FOR TRANSMITS)         - THISJES
            AUTOMATIC DIRECTION IS ALLOWED
            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
                  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
            FULL RRSF COMMUNICATION IS *NOT* ENABLED
            RACF STATUS INFORMATION:
                  TEMPLATE VERSION          - HRF77A0  00000194.00000022
                  DYNAMIC PARSE VERSION     - HRF77A0
```

Figure 11. Sample output from the SET LIST command

Tracing APPC, IMAGE, and RRSF events

You can obtain trace records for IMAGE, APPC, and RRSF events using the Generalized Trace Facility (GTF). RRSF events include, but are not limited to, communication-related events using TCP/IP. 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, APPC, or RRSF 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 or RRSF events, enter:

```
SET TRACE(APPC NOIMAGE NORRSF)
```

To continue tracing APPC events, and to turn on IMAGE tracing, enter:

```
SET TRACE(IMAGE)
```

Because no setting for APPC or RRSF events is specified, the current settings remain 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 and password phrase changes are to be synchronized
- Which users' password and password phrase changes are automatically directed and to which remote nodes

If the RRSFDATA class is not active, no password or password phrase 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 by way of 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.

Note: The TSO SEND command updates the broadcast data set. This can either be a single data set used for all users, such as SYS1.BRODCAST, or a user-specific data set. The use of a global data set like SYS1.BRODCAST can result in heavy contention and deadlocks within the RACF subsystem address space, and even across systems in a sysplex. See the IKJTSOxx documentation in *z/OS MVS Initialization and Tuning Reference* for information on defining individual user logs.

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 about 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 188.

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. If you specify the JESNODE keyword, the node name you specify overrides the node name obtained from the query.

Allocating workspace files for remote non-MAIN connections in preparation for dynamic main switches

By default, a non-MAIN system within a given multisystem node will not communicate with any non-MAIN system of a different multisystem node. Their state in relation to each other is DEFINED, and there are no workspace files or network connections between them. However, if you plan to use the dynamic MAIN switch function provided by the TARGET NEWMAIN or TARGET PLEXNEWMAIN commands, the new MAIN system will need to communicate with every remote system in the RRSF network. For information on performing dynamic MAIN switches, see “Configuring a new main system in a multisystem node” on page 175.

To allow remote non-MAIN systems to communicate with each other, use the SET FULLRRSFCOMM command. When SET FULLRRSFCOMM is in effect, the DORMANT and OPERATIVE keywords of the TARGET command will behave the same way for any remote system, including the case where both the local and remote systems are non-MAIN members of a multisystem node. In brief, DORMANT and OPERATIVE will allocate workspace files for the remote target if they do not already exist, and OPERATIVE will, in addition, establish a network connection. You must enable SET FULLRRSFCOMM and make connections between non-MAIN systems OPERATIVE before you can successfully use the

PLEXNEWMAIN or NEWMAIN keyword of the TARGET command to perform a dynamic MAIN switch. This must be performed on every non-MAIN system of every multisystem node in the network. Be aware that enabling FULLRRSFCOMM will result in the allocation of new workspace files and network connections even when they are not being used.

If you never intend to switch MAIN systems dynamically, then you should not enable FULLRRSFCOMM. SET FULLRRSFCOMM is not required on any system in a multisystem node if it is the only multisystem node in the network. It is never required on a single system node.

Consider placing the SET FULLRRSFCOMM command in the RACF parameter library, prior to the TARGET commands which will define the RRSF network, well in advance of needing to perform a dynamic MAIN switch (possibly even before IPLing z/OS V2R2 for the first time). As each system sharing the parameter library is IPLed, it will be prepared to become the MAIN system at any time. On any V2R1 level or lower system (running a release prior to z/OS V2R2), the SET FULLRRSFCOMM command will simply fail, harmlessly, with a syntax error. When a V2R1 level or lower system is upgraded to V2R2 and IPLed for the first time, it will process the command.

However, a re-IPL is not required to use FULLRRSFCOMM. You can issue SET FULLRRSFCOMM on the operator console of a live non-MAIN system, and then issue the TARGET commands to make all remote non-MAIN connections DORMANT (to allocate the files) or OPERATIVE (to allocate the files and establish a connection). For example, from each non-MAIN system in the multisystem node, issue:

```
    SET FULLRRSFCOMM
    TARGET NODE(msn-1) SYSNAME(*) OPERATIVE
    TARGET NODE(msn-2) SYSNAME(*) OPERATIVE
```

In this case, you must also put the SET FULLRRSFCOMM command in the RACF parameter library so that it gets processed on the next IPL.

A successful connection from a locally enabled non-MAIN system to a remote non-MAIN system will depend on whether the remote system also has files allocated on its side of the connection. If a remote system is V2R1 level or lower, or does not have FULLRRSFCOMM in effect, a message is issued to the local console and the connection will be left in the DORMANT-REMOTE state:

```
IRRC059I (>) RACF REMOTE SHARING CONNECTION TO NODE node-name
 SYSNAME system-name DID NOT COMPLETE SUCCESSFULLY. REMOTE NODE
 REPORTS ITS STATE AS DEFINED.
```

For an APPC connection, the following message is issued to the log (not to the console):

```
IRRC024I (>) RACF REMOTE SHARING CONNECTION TO NODE node-name
 SYSNAME system-name DID NOT COMPLETE SUCCESSFULLY.
 FAILURE OCCURRED WHEN APPC VERB ATBSEND WAS ISSUED.
 RETURN CODE = 22.
```

On the remote system, the connection will remain in the DEFINED state. If the remote system is at the V2R2 level or higher, but disabled, then when it is enabled, and made OPERATIVE from that side of the connection, the connection will be established.

If the remote system is V2R1 level or lower, then the local system will essentially always be in DEFINED state from the remote system's perspective, until that remote system is upgraded and enabled. Be aware that locally, the files have been allocated in either case, and thus they must be accounted for during the capacity planning that should occur prior to enabling FULLRRSFCOMM.

If you remove the SET FULLRRSFCOMM command from the parameter library after the files have already been allocated, they will not be automatically deleted. You can issue TARGET DORMANT followed by TARGET DELETE and, assuming the files are empty, they will be automatically deleted by TARGET DELETE. The node or system will get redefined into the DEFINED state on the subsequent IPL. Alternatively, if you simply IPL again, the connection will be in the DEFINED state, and you can manually delete the files because they will not be in use.

Defining RRSF nodes to RACF using 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*

TARGET is an operator command, and it can be issued in three different ways:

- An operator can enter the command manually, at the console.
- An operator can enter a SET INCLUDE(*xx*) command at the console, where *xx* specifies a member of the RACF parameter library that issues a TARGET command, or includes another member that does.
- RACF subsystem initialization can automatically invoke a member of the RACF parameter library that invokes a TARGET command, or includes another member that does. The member is specified as a parameter in the RACF PROC.

For 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*.

A z/OS system image must meet the following requirements to be defined as an RRSF node:

- The RACF component of the z/OS Security Server is enabled.
- The RACF subsystem address space is active.

TARGET commands are cumulative. That is, you can specify a subset of the configuration information for a node on one TARGET command, and additional information for the same node on subsequent TARGET commands.

- For APPC connections, once you specify OPERATIVE or DORMANT on a TARGET command for a node, RACF activates the node or makes it dormant, and any further TARGET commands that attempt to change the specification of the protocol or workspace data sets for a node are rejected. To change the protocol or workspace data set specification, you must delete the node and then redefine it.
- For TCP/IP connections, once you specify OPERATIVE on a TARGET command for a node, RACF activates the node and any further TARGET commands that attempt to change the specification of the protocol or workspace data sets for a node are rejected. However, you can change the protocol information if the node is dormant.

Tip: When defining or changing an RRSF network, first test the TARGET commands by entering them manually at the console. When you are satisfied that they are correct, add them to a RACF parameter library member that is invoked by RACF subsystem initialization, so that the changes persist over system IPLs or if the RACF subsystem address space is stopped and restarted. Keep in mind that you will probably need to make corresponding changes to parameter library members on remote nodes. It is strongly recommended that systems in a multisystem node use a shared RACF parameter library member.

You must specify a node name, workspace information, and the high level qualifiers to be used on the INMSG and OUTMSG data sets for any node, local or remote, before you can make the node operative. You must define protocol information for the local node and the local node must be in an operative state before you can make connections to remote nodes operative. Protocol information must also be defined for a remote node before you can make a remote connection operative. In addition, if the remote 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 (for APPC only)
- Node names
- System names

- Protocol
- 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 all use APPC and 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. (For TCP/IP connections, RACF issues message IRRI016I instead of IRRI014I.)

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.

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 163 for information on when the LIST keyword is defaulted to.) If SYSNAME is not specified, and LIST is not specified or used as the default, RACF assumes that the node is a single-system node.

The value of SYSNAME 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. Because it is used as a data set qualifier for the local node's workspace data sets, SYSNAME should not have a numeric as the first character. If the value of 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 node's workspace data set name.

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
- ALLOWINBOUND
- DENYINBOUND
- RESETDENYINBOUNDCOUNT

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 DENYINBOUND keyword for a remote node to define a one-way outbound connection to that node. ALLOWINBOUND is the default for this setting. Both of these keywords are ignored for the local node.

When DENYINBOUND is specified for a given node, incoming work from that system will be discarded by the receiving system. This will not interfere with the ability to send work to the targeted node and get returned output and notifications from the work sent to the targeted node.

When specifying ALLOWINBOUND or DENYINBOUND for a system within a multisystem node, it will be automatically applied to all systems in that node. Nonetheless, when specifying DENYINBOUND in a RACF parameter library member, it is recommended that you specify it for all systems in the multisystem node.

RESETDENYINBOUNDCOUNT resets the count of denied inbound work requests to 0 for the node and system specified on the command. This count is maintained by RRSF and displayed by the TARGET LIST command. The count can be used to identify systems that are inappropriately configured to send updates.

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 issue 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. When either the local or remote system is lower than z/OS V2R2, 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 remote multisystem node before you make any systems in the multisystem node operative.

The NEWMAIN and PLEXNEWMAIN operational keywords can be used to dynamically change the MAIN system. See "Configuring a new main system in a multisystem node" on page 175.

Because the main system in a multisystem node receives the majority of RRSF traffic to the node, be sure to select a main 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.

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 is 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, and to specify the network transport mechanism to be used. Specify either APPC or TCP. The subkeywords of PROTOCOL specify information about the network transport mechanism. 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, unnecessary processing occurs.

If you specify APPC for the protocol, use the LUNAME subkeyword on the PROTOCOL(APPC()) 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.

If you specify APPC for the protocol, use the optional TPNAME subkeyword on the PROTOCOL(APPC()) keyword to identify the APPC transaction program (TP) profile. The TP profile name is 1 - 64 characters in length, and defaults to IRRRACF. Once you have specified a TP profile name for a node, you can change it only if you first make the node dormant.

Guideline: Let RACF take the default and use IRRRACF as the TPNAME unless you need to change it.

If you specify APPC for the protocol, use the optional MODENAME subkeyword on the PROTOCOL(APPC()) 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 - 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.

If you specify TCP for the protocol, use the ADDRESS subkeyword on the PROTOCOL(TCP()) keyword to identify the remote system. Specify either the host name of the target system or its static IP address. ADDRESS is not required for the local node, and if not specified, RACF uses the default value of 0.0.0.0. ADDRESS must be specified for a remote node before a TARGET command making it operative is issued. An IP address may be specified as an IPv4 address or an IPv6 address (if TCP IPv6 is enabled on the system). See *z/OS Communications Server: IPv6 Network and Appl Design Guide* for more information about the IPv6 protocol in z/OS Communication Server.

If you specify TCP for the protocol, use the optional PORTNUM subkeyword on the PROTOCOL(TCP()) keyword to specify the port on which the RRSF node will establish a socket in order to listen for requests initiated by a remote node. If not specified, the default value is 18136.

Guideline: Use the default value for PORTNUM.

Use the PREFIX keyword to specify one or more high-level qualifiers for the data set names of the INMSG and OUTMSG workspace data sets. 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.

Guideline: Specify the same prefix for all systems on a multisystem node, to allow you to reconfigure the multisystem node with a different main system using the instructions in Appendix C, “MAIN switch procedure in a mixed-release network,” on page 359. This requirement does not apply to dynamic MAIN switches using the PLEXNEWMAIN and NEWMAIN keywords.

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 for an APPC connection, or to the name specified on the NODE keyword for a TCP/IP connection to a single-system node, or to the name specified on the SYSNAME keyword for a TCP/IP

connection to a multisystem node. For an APPC connection, 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.

If you specify the WDSQUAL keyword, you are responsible for ensuring that the resulting workspace data set names do not conflict with the workspace data set names for any other nodes. For more information on the workspace data sets, see "Workspace data sets" on page 136.

Use the WORKSPACE keyword to specify the attributes of the workspace data sets. 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 136.

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. **Guideline:** Create the workspace data sets as SMS-managed data sets that can grow as needed.

- 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.

Guideline: For multisystem nodes, 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 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 - 2147483647. The initial value is FILESIZE(500). For some guidelines on what size to specify, see "Size guidelines for the workspace data sets" on page 139.

Listing the attributes of target nodes

Use the LIST keyword or the LISTPROTOCOL 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 and LISTPROTOCOL keywords also display information about the systems that make up the node. If you specify the NODE keyword with LIST or LISTPROTOCOL, the output shows detailed information for the node. If you do not specify the NODE keyword, the output shows summary information for all nodes in the network. The output for the LIST and LISTPROTOCOL keywords is the same when you also specify the NODE keyword. If you do not specify the NODE keyword, the LIST keyword only lists protocol information for nodes that have multiple protocols defined, but the LISTPROTOCOL keyword lists protocol information for all nodes.

Tip: Use LISTPROTOCOL when you have a mixed protocol network and you want to quickly identify the protocol used by each node without displaying detailed information for each node.

You can request detailed information for one target node or for all target nodes by specifying the NODE keyword. You can specify the PROTOCOL and SYSNAME keywords with the NODE keyword to qualify the information that you want displayed. For example, specify `TARGET LIST NODE(*) PROTOCOL(TCP)` to request detailed information about all nodes that use the TCP/IP protocol. For information about the combinations of keywords that you can specify and the output that results, see *z/OS Security Server RACF Command Language Reference*. 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

"Defining the workspace data sets" on page 138 for information about why the values shown might not be the values in effect.

When using IPv6 on z/OS, then the TARGET LIST command for a node displays a resolved address in IPv6 format. This also occurs when an IPv4 address is specified for the ADDRESS operand after the address is resolved with an OPERATIVE connection.

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 12 on page 164 illustrates the summary information displayed for a TARGET LIST command.

```
IRRM009I (<) LOCAL RRSF NODE NODE1 IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (<)      - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM091I (<)      - LOCAL NODE APPC LISTENER IS ACTIVE.
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 12. Summary information displayed by a TARGET LIST command for a single-system node

Figure 13 on page 165 illustrates the detailed information displayed for a TARGET LIST command for a node using APPC/MVS.

```
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>
                 LISTENER STATUS  - ACTIVE
   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 13. Detailed information displayed by a TARGET LIST command for a single-system node using APPC/MVS.

Figure 14 on page 166 illustrates the detailed information displayed for a TARGET LIST command for a node using TCP/IP. In this output:

- The IP ADDRESS line is displayed only if the resolved IP address differs from the host address specified on the TARGET command for this node. The IP address value shown is the value determined the last time a remote connection was attempted.
- For operative connections, some (but not all) attributes from the AT-TLS policy are shown. Use the NETSTAT command to see additional information. For information about the NETSTAT command, see *z/OS Security Server RACF Diagnosis Guide* and *z/OS Communications Server: IP System Administrator's Commands*.
- A MAPPED USER line is displayed for the AT-TLS policy if the client authentication level is SAFCHECK. It displays the user ID to which the connecting system's digital certificate is mapped.
- The rule name and client authentication are always displayed in uppercase, even if they are in mixed case in the AT-TLS policy.
- The cipher is displayed as a number followed by a string. The numbers and string values for the cipher suites are defined by the relevant SSL or TLS RFCs. The values supported by System SSL are documented in the description of the `gsk_environment_open()` service in *z/OS Cryptographic Services System SSL Programming*. The values supported by AT-TLS are documented in the description of the `TTLSCipherParms` statement in *z/OS Communications Server: IP Configuration Reference*. The value shown in the figure is documented by System SSL to mean "256-bit AES encryption with SHA-1 message authentication and RSA key exchange".

```
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
   STATE - OPERATIVE ACTIVE
   DESCRIPTION - <NOT SPECIFIED>
   PROTOCOL - TCP
              HOST ADDRESS - MVS5.POK.OURS.COM
              IP ADDRESS - 9.57.1.13
              LISTENER PORT - 18136
              AT-TLS POLICY:
                RULE_NAME - RRSF-CLIENT
                CIPHER ALG - 35 TLS_RSA_WITH_AES_256_CBC_SHA
                CLIENT AUTH - REQUIRED
   TIME OF LAST TRANSMISSION TO - 16:45:39 DEC 15, 2010
   TIME OF LAST TRANSMISSION FROM - 16:45:40 DEC 15, 2010
   WORKSPACE FILE SPECIFICATION
         PREFIX - "SYS1.RRSF"
         WDSQUAL - <NOT SPECIFIED>
         FILESIZE - 500
         VOLUME - DASD01
         FILE USAGE
                 "SYS1.RRSF.NODE1.NODE2.INMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
                 "SYS1.RRSF.NODE1.NODE2.OUTMSG"
                                  - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
```

Figure 14. Detailed information displayed by a TARGET LIST command for a single-system node using TCP/IP.

Figure 15 on page 166 illustrates the detailed information displayed for a TARGET LIST NODE(*nodename*) command for a local node that uses both TCP/IP and APPC/MVS.

```
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF LOCAL RRSF NODE NODE1:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - <NOT SPECIFIED>
   PROTOCOL    - TCP
                 HOST ADDRESS      - 0.0.0.0
                 IP ADDRESS        - 9.57.1.243
                 LISTENER PORT     - 18136
                 LISTENER STATUS   - ACTIVE
   PROTOCOL    - APPC
                 LU NAME           - MF1AP001
                 TP PROFILE NAME   - IRRRACF
                 MODENAME          - <NOT SPECIFIED>
                 LISTENER STATUS   - ACTIVE
   TIME OF LAST TRANSMISSION TO    - <NONE>
   TIME OF LAST TRANSMISSION FROM  - <NONE>
   WORKSPACE FILE SPECIFICATION
         PREFIX                    - "SYS1.RRSF"
         WDSQUAL                   - <NOT SPECIFIED>
         FILESIZE                  - 500
         VOLUME                    - TEMP01
         FILE USAGE
                "SYS1.RRSF.NODE1.INMSG"
                                   - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
                "SYS1.RRSF.NODE1.OUTMSG"
                                   - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
```

Figure 15. Detailed information displayed by a TARGET LIST NODE(nodename) command for a single-system local node that uses both APPC/MVS and TCP/IP.

Figure 16 on page 167 illustrates the information displayed for a TARGET LISTPROTOCOL command. Figure 17 on page 167 illustrates the information displayed for a TARGET LIST command issued on the same system. This output is the same as the TARGET LISTPROTOCOL output except that it does not list the protocol for each remote node.

```
IRRM009I (<) LOCAL RRSF NODE NODE1 IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (<)      - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM091I (<)      - LOCAL NODE APPC LISTENER IS ACTIVE.
IRRM009I (<) REMOTE RRSF NODE NODE2 PROTOCOL APPC IS IN THE OPERATIVE
ACTIVE STATE.
IRRM009I (<) REMOTE RRSF NODE NODE3 PROTOCOL APPC IS IN THE OPERATIVE
PENDING CONNECTION STATE.
IRRM009I (<) REMOTE RRSF NODE NODE4 PROTOCOL TCP IS IN THE OPERATIVE
PENDING CONNECTION STATE.
IRRM009I (<) REMOTE RRSF NODE NODE5 PROTOCOL TCP IS IN THE OPERATIVE
PENDING CONNECTION STATE.
```

Figure 16. Information displayed by a TARGET LISTPROTOCOL command.

```
IRRM009I (<) LOCAL RRSF NODE NODE1 IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (<) - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM091I (<) - LOCAL NODE APPC LISTENER IS ACTIVE.
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 NODE5 IS IN THE OPERATIVE PENDING
CONNECTION STATE.
```

Figure 17. Information displayed by a TARGET LIST command.

Sample output for multisystem nodes that use APPC/MVS

Figure 18 on page 167 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 (MAIN) IS IN THE OPERATIVE
  -  ACTIVE STATE.
  - IRRM091I (@) - LOCAL NODE APPC LISTENER IS ACTIVE.
  - 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 18. Summary information from the TARGET LIST command for a multisystem node

Figure 19 on page 168 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 (MAIN)
  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 19. Detailed information from the TARGET LIST command for a system on a multisystem node

Sample output for multisystem nodes that use TCP/IP

Figure 20 on page 168 shows an example of the summary information that might be displayed for a multisystem node named NODEABC. It contains three systems named MVSA, MVSB and MVSC. The TARGET LIST command was issued from MVSA, which is the main system for NODEABC.

```
IRRM009I (<) LOCAL RRSF NODE NODEABC SYSNAME MVSA (MAIN) IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (<) - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM009I (<) LOCAL RRSF NODE NODEABC SYSNAME MVSB IS IN
THE DEFINED STATE.
IRRM009I (<) LOCAL RRSF NODE NODEABC SYSNAME MVSC IS IN THE DEFINED STATE.
```

Figure 20. Summary information from the TARGET LIST command for a multisystem node that uses TCP/IP

Figure 21 on page 169 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(NODEABC) SYSNAME(MVSA) LIST was issued from system MVSA on node NODEABC.

```
IRRM010I (<) RSHJ SUBSYSTEM PROPERTIES OF LOCAL RRSF NODE NODEABC
SYSNAME MVSA (MAIN):
  STATE  - OPERATIVE ACTIVE
  DESCRIPTION - <NOT SPECIFIED>
  PROTOCOL    - TCP
       HOST ADDRESS                - 0.0.0.0
       IP ADDRESS                  - 9.57.1.145
       LISTENER PORT               - 18136
       LISTENER STATUS             - ACTIVE
  TIME OF LAST TRANSMISSION TO     - <NONE>
  TIME OF LAST TRANSMISSION FROM   - <NONE>
  WORKSPACE FILE SPECIFICATION
        PREFIX                     - "RSF1"
        WDSQUAL                    - <NOT SPECIFIED>
        FILESIZE                   - 500
        VOLUME                     - TEMP01
        FILE USAGE
            "RSF1.MVSA.INMSG"
                                   - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
            "RSF1.MVSA.OUTMSG"
                                   - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
```

Figure 21. Detailed information from the TARGET LIST command for a system on a multisystem node

Sample output for local nodes that use IPv6

Figure 22 on page 170 shows an example of the local node from a TARGET LIST command when IPv6 is enabled and no ADDRESS is specified in the local node definition.

```
<target list node(node1)
IRRM010I (<) RSWJ 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>
                 LISTENER STATUS  - ACTIVE
   PROTOCOL    - TCP
                 HOST ADDRESS     - ::                       <<< IPV6 default
                 IP ADDRESS       - ::FFFF:9.57.1.243        <<< IPv6 address
                 LISTENER PORT    - 18136
                 LISTENER STATUS  - ACTIVE
   TIME OF LAST TRANSMISSION TO   - <NONE>
   TIME OF LAST TRANSMISSION FROM - <NONE>
   WORKSPACE FILE SPECIFICATION
         PREFIX                   - "SYS1.RRSF"
         WDSQUAL                  - <NOT SPECIFIED>
         FILESIZE                 - 500
         VOLUME                   - TEMP01
         FILE USAGE
                 "SYS1.RRSF.NODE1.INMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
                 "SYS1.RRSF.NODE1.OUTMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
```

Note:

`IP ADDRESS - ::FFFF:9.57.1.243`**:** When IPv6 is enabled, the resolved address is displayed in IPv6 format.

If IPv6 is not enabled or the command is issued on z/OS before V2R1, the default HOST ADDRESS shows 0.0.0.0 and the IP address shows as 9.57.1.243.

Figure 22. Example from the TARGET LIST command for local nodes that use IPv6, no address specified

Figure 23 on page 171 shows an example detailed TARGET LIST output for a remote TCP/IP node when TCP/IP IPv6 is enabled. A host name is specified on the ADDRESS operand.

```
<target list node(node2)
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - <NOT SPECIFIED>
   PROTOCOL    - TCP
                 HOST ADDRESS     - MVS8.POK.OUT.COM   <<< entered on TARGET
                 IP ADDRESS       - 2001:0db8:85a3:0000:0000:8a2e:0370:7334
                 LISTENER PORT    - 18136
                 AT-TLS POLICY:
                   RULE_NAME      - RRSF-CLIENT
                   CIPHER ALG     - 35 TLS_RSA_WITH_AES_256_CBC_SHA
                   CLIENT AUTH    - REQUIRED
   TIME OF LAST TRANSMISSION TO   - <NONE>
   TIME OF LAST TRANSMISSION FROM - <NONE>
   WORKSPACE FILE SPECIFICATION
         PREFIX                   - "SYS1.RRSF"
         WDSQUAL                  - <NOT SPECIFIED>
         FILESIZE                 - 500
         VOLUME                   - TEMP01
         FILE USAGE
                 "SYS1.RRSF.NODE1.NODE2.INMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
                 "SYS1.RRSF.NODE1.NODE2.OUTMSG"
                                  - CONTAINS 0 RECORD(S)
```

Note:

`IP ADDRESS - 2001:0db8:85a3:0000:0000:8a2e:0370:7334`**:** When IPv6 is enabled, the resolved address is displayed in IPv6 format.

Figure 23. Detailed information from the TARGET LIST command output for a remote TCP/IP node when TCP/IP IPv6 is enabled, host name is specified

Figure 24 on page 172 shows an example detailed TARGET LIST output for a remote TCP/IP node when TCP/IP IPv6 is enabled. An IPv4 address is specified on the ADDRESS operand.

```
<target list node(node2)
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
   STATE       - OPERATIVE ACTIVE
   DESCRIPTION - <NOT SPECIFIED>
   PROTOCOL    - TCP
                 HOST ADDRESS     - 10.120.78.40       <<< entered on TARGET
                 IP ADDRESS       - ::FFFF:10.120.78.40 <<< IPv6 address
                 LISTENER PORT    - 18136
                 AT-TLS POLICY:
                   RULE_NAME      - RRSF-CLIENT
                   CIPHER ALG     - 35 TLS_RSA_WITH_AES_256_CBC_SHA
                   CLIENT AUTH    - REQUIRED
   TIME OF LAST TRANSMISSION TO   - <NONE>
   TIME OF LAST TRANSMISSION FROM - <NONE>
   WORKSPACE FILE SPECIFICATION
         PREFIX                   - "SYS1.RRSF"
         WDSQUAL                  - <NOT SPECIFIED>
         FILESIZE                 - 500
         VOLUME                   - TEMP01
         FILE USAGE
                 "SYS1.RRSF.NODE1.NODE2.INMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
                 "SYS1.RRSF.NODE1.NODE2.OUTMSG"
                                  - CONTAINS 0 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
```

Note:

`IP ADDRESS - ::FFFF:10.120.78.40`**:** When IPv6 is enabled, the resolved address is displayed in IPv6 format.

If IPv6 is not enabled or the command is issued on z/OS before V2R1, the IP ADDRESS line does not show because the resolved address is the same as the HOST ADDRESS line.

Figure 24. Detailed information from the TARGET LIST command output for a remote TCP/IP node when TCP/IP IPv6 is enabled and an IPv4 address is specified

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.

Note: This section assumes the target is accepting inbound work from the local node. The DENYINBOUND keyword of the TARGET command defines a node from which inbound work is not accepted. If the target node defines the local node in this way, any outbound updates sent from the local node will be rejected at the target node.

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 remote node (such as directed commands) that are held in the OUTMSG data set are sent to the node. As long as the connection remains operative, new requests for the node are sent when they are placed in the OUTMSG data set.

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 "Connection states" 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 saves requests for the remote node (such as directed commands) in the OUTMSG workspace data set for the node. The requests are held until the connection is made operative. Requests that have been received from the remote node, but not yet processed, are saved in the INMSG data set. RACF continues to process requests in the INMSG data set even if the connection is dormant.

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

Occasionally it is desirable to set up RRSF such that updates made on a local system get propagated outbound to other systems, but to prevent updates made on some set of the remote systems from being propagated back to the local system. While you cannot prevent the other system from trying to send you requests, the DENYINBOUND keyword of the TARGET command would enable a local system the ability to refuse an incoming request by creating a one-way outbound connection to a node on the network.

For more information about DENYINBOUND and ALLOWINBOUND kewords, see the discussion in "Defining RRSF nodes to RACF using the TARGET command" on page 159.

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 in the OUTMSG data set of the remote node are sent to the local node. As long as the connection remains operative, new requests from the remote node are sent when they are placed in the OUTMSG data set.

Use the DORMANT keyword to request that a connection to a node be made inactive, or dormant. When a connection is dormant, RACF saves requests from the remote node (such as directed commands) in its OUTMSG workspace data set on the remote node. The requests are held until the connection is made operative. Requests that have been received from the remote node, but not yet processed, are saved in the INMSG data set. RACF continues to process requests in the INMSG data set even if the connection is dormant.

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.

Rules: There is an order you need to observe when deleting nodes. The TARGET command enforces these rules:

- 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

Please review and understand the following terms before you read this topic.

Original MAIN
The system in the multisystem node designated as main by the first TARGET commands issued to define the RRSF network. In practice, these commands are issued automatically from within the RACF parameter library member specified by the RACF subsystem JCL.

Old MAIN
The system in the multisystem node that was main prior to a dynamic main switch.

New MAIN
The system in the multisystem node which becomes main after a dynamic main switch.

Ex-MAIN
The old MAIN, from the perspective of a remote target, until such time as a new MAIN is established.

Peer system
One of the systems in a multisystem node which is neither the old MAIN nor the new MAIN during a dynamic main switch.

Nonmain
One of the systems in a multisystem node which is not the main system.

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. The PLEXNEWMAIN and NEWMAIN operands of the TARGET command allow you to perform a dynamic main switch without losing any RRSF requests or returned output. RRSF ensures that requests execute in the proper order.

- Use the PLEXNEWMAIN operand when running in a sysplex environment. It is the simplest option, and maximally reduces the risk of manual error. With PLEXNEWMAIN, you perform the switch by issuing a single TARGET command from any one system in the multisystem node.
- Use the NEWMAIN operand when not running in a sysplex environment. With NEWMAIN, you issue a single TARGET command on each system in the multisystem node, and the order and timing of the commands is important. NEWMAIN also works in a sysplex environment, and might be helpful to recover from PLEXNEWMAIN failures.

Attention: The procedures documented in this section can only be used if both the current and new MAIN systems are at z/OS V2R2 or higher; it is acceptable if other peer systems are running a lower release. In addition, if there are lower than V2R2 remote systems in your RRSF network, the switch must only be in effect for a temporary interval. If the current or new MAIN system is lower than V2R2, or if you wish to perform a permanent switch when there are lower than V2R2 remote systems in your RRSF network, follow the instructions documented in Appendix C, “MAIN switch procedure in a mixed-release network,” on page 359.

Note:

1. PLEXNEWMAIN and NEWMAIN cannot be used unless SET FULLRRSFCOMM was in effect prior to remote connections being established. See “Allocating workspace files for remote non-MAIN connections in preparation for dynamic main switches” on page 157 for details.
2. PLEXNEWMAIN has no dependency on the use of data sharing mode or sysplex communication mode to share the RACF database. A completely independent set of XCF communication services is used.

A dynamic main switch essentially automates the following procedure on the switching node:

1. The old (current) main system puts all of its operative remote connections into DORMANT state.
2. The old MAIN updates its main status.

3. The old MAIN completes any work it has in progress. It monitors all INMSG files for remote connections until they are empty. The amount of time required for all INMSG files to drain depends on the amount of work contained within them and on how busy the system is. This could take as little as several seconds, or as much as 30 minutes, after which the switch will fail with message IRRM100I.
4. When the INMSG file for a given connection is empty, the old MAIN reestablishes that connection if it was made DORMANT as part of the switch process. Output is returned as appropriate. This step is considered complete only after all remote connections have been processed.
5. The new MAIN system puts all of its operative remote connections into DORMANT state.
6. The new MAIN updates its main status.
7. The new MAIN reestablishes the connections it made DORMANT.
8. The other peer systems within the switching node update their main status.

The change in main status is communicated to the remote targets as part of the handshaking process that occurs during connection establishment. This requires each connection to be broken and then reestablished, resulting in a number of informational messages being issued to the console and system log during the switch. Note that no commands need to be issued on remote nodes.

From the perspective of a given remote partner, the following sequence of events occurs:

1. The old (current) MAIN system makes its connection DORMANT.
2. The old MAIN makes its connection OPERATIVE and its handshaking data indicates it is not the MAIN system. This old main system is displayed as “EX-MAIN” in TARGET LIST output. Outbound work for the switching node continues to be checkpointed in the OUTMSG file for the EX-MAIN system, but is not sent to it.
3. The new MAIN makes its connection DORMANT.
4. The new MAIN makes its connection OPERATIVE and its handshaking data indicates it is the main system. Local status is updated to indicate the new main system, and to clear the EX-MAIN status of the previous MAIN.
5. All OUTMSG files for all systems in the switching node are inspected for checkpointed work, which is sent to the new MAIN in the proper order. TARGET LIST displays all such files under consideration in a section with the heading “SCANNING FILE(S)”. An OUTMSG file is removed from consideration when RRSF detects that no more work remains within it. Eventually, the only file used is the regular OUTMSG file for the new main system, and the “SCANNING FILE(S)” section is no longer displayed.

Exception: A RACLINK command is always sent to the main system in effect at the time the command is issued. If the connection to that system is DORMANT when the command is issued, it is checkpointed and sent to that system as soon as its connection is reestablished. That system will process the RACLINK command normally, even if it is no longer the main system for that node.

A dynamic MAIN switch is only recognized by a remote target running z/OS V2R2 or a higher. On a lower release, a handshaking error results from the mismatch in MAIN status when both the old and new MAIN system reconnect. The connection is not reestablished (it stays in OPERATIVE PENDING VERIFICATION state). On the remote target, work continues to be checkpointed in the OUTMSG file for the old MAIN system until it fills up, or until the old MAIN connects as MAIN again. Likewise, outbound work initiated on both the old and new MAIN system will get checkpointed in the OUTMSG file for the remote target. Therefore, as long as you have lower than V2R2 systems in your RRSF network, you should make the MAIN switch as temporary as possible.

Performing a MAIN switch in a sysplex environment

Rule: The current and new MAIN systems must be at z/OS V2R2 or higher to use this procedure.

In a sysplex, XCF services are used to coordinate the switch among peer systems in a multisystem node. In this environment, almost the entire process is accomplished with a single command.

1. Issue **TARGET** PLEXNEWMAIN

From any system in the multisystem node that is at z/OS V2R2 or higher, issue a **TARGET** PLEXNEWMAIN command specifying the new MAIN system. For example:

```
TARGET NODE(THISMSN) SYSNAME(NEXTMAIN) PLEXNEWMAIN
```

When processing is complete, IRRM110I is issued on all V2R2 and higher systems in the multisystem node to indicate completion of the switch. For example, if the old MAIN system is OLDMAIN:

```
IRRM110I SYSTEM NEXTMAIN HAS REPLACED SYSTEM OLDMAIN AS THE MAIN SYSTEM IN LOCAL NODE
THISMSN.
```

2. Update lower than V2R2 peer systems

   If there are any peer systems within the multisystem node running lower than z/OS V2R2, the updated status cannot be automatically communicated to them.

   During the PLEXNEWMAIN command, you will receive messages IRRC077I and IRRM109I for each of those systems. For example, if you are changing the main from OLDMAIN to NEXTMAIN in node THISMSN and systems PEER1 and PEER2 (also in THISMSN) are running z/OS V2R1, you would see the following messages.

   ```
   TARGET PLEXNEWMAIN NODE(THISMSN) SYSNAME(NEXTMAIN)
   IRRM002I RSAJ SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
   IRRC077I RSAJ SUBSYSTEM DID NOT RECEIVE AN EXPECTED RESPONSE FROM SYSTEM PEER1.
   IRRM109I RSAJ SUBSYSTEM COULD NOT NOTIFY SYSTEM PEER1 THAT SYSTEM NEXTMAIN IS
                NOW THE MAIN SYSTEM IN LOCAL NODE THISMSN.
   IRRC077I RSAJ SUBSYSTEM DID NOT RECEIVE AN EXPECTED RESPONSE FROM SYSTEM PEER2.
   IRRM109I RSAJ SUBSYSTEM COULD NOT NOTIFY SYSTEM PEER2 THAT SYSTEM NEXTMAIN IS
                NOW THE MAIN SYSTEM IN LOCAL NODE THISMSN.
   IRRM110I SYSTEM NEXTMAIN HAS REPLACED SYSTEM OLDMAIN AS THE MAIN SYSTEM IN LOCAL
                NODE THISMSN.
   ```

   In this case, the main switch was successful, but since PEER1 and PEER2 are lower than z/OS V2R2, the main switch is not reflected on those systems. Until this step is performed, a **TARGET** LIST command issued on such a system displays the old MAIN as the main system. To update these systems, issue a **TARGET** MAIN command on both PEER1 and PEER2. For example:

   ```
   TARGET MAIN NODE(THISMSN) SYSNAME(NEXTMAIN)
   ```

3. Update the RACF parameter library

   When an V2R2 or higher system IPLs, it queries an V2R2 or higher peer system to obtain the name of the current MAIN system. If there are no such systems available (for example, this is the first system being IPLed in a sysplex-wide IPL), the MAIN system coded in the parameter library will be used.

   If you have lower than V2R2 systems in your sysplex, they rely on the accuracy of the parameter library to establish the intended main system, so it is especially important to update the parameter library in this case.

   Note: Update TARGET commands by moving the MAIN keyword from the TARGET statement of the original MAIN of the local node to that of the new MAIN.

Performing a MAIN switch in a non-sysplex environment

Rule: The current and new MAIN systems must be at z/OS V2R2 or higher to use this procedure.

In a non-Sysplex environment, peer systems do not communicate with each other in order to coordinate the switch automatically. The **TARGET** NEWMAIN command must be issued on each system in a specific sequence.

1. Initiate the switch on the current main system.

   In order to initiate the switch, issue a **TARGET** NEWMAIN command specifying the new MAIN system. For example:

   ```
   TARGET NODE(THISMSN) SYSNAME(NEXTMAIN) NEWMAIN
   ```

The **TARGET** command warns the user not to continue immediately to step 2:

```
IRRM098I DRAINING SYSTEM OF INBOUND WORK. DO NOT INITIATE THE MAIN SWITCH ON THE
         NEW MAIN SYSTEM UNTIL MESSAGE IRRM099I IS ISSUED.
IRRM002I RSAJ SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
```

When the old MAIN is finished processing, it issues message IRRM099I:

```
IRRM099I All INBOUND WORK HAS COMPLETED. IT IS NOW SAFE TO
         INITIATE THE MAIN SWITCH ON THE NEW MAIN SYSTEM.
```

Until the next step is performed, there will be no main system in the switching multisystem node. The remote systems' OUTMSG files to the old main system are used to checkpoint work until a new main system is recognized. Thus it is important to perform the following step without delay, so that no OUTMSG file becomes full.

2. Complete the switch on the new main system.

   Attention: Wait for message IRRM099I to be issued on the old main system before performing this step, or work might be run out of order.

   On the system intended to be the new MAIN, issue the same **TARGET** command:

   ```
   TARGET NODE(THISMSN) SYSNAME(NEXTMAIN) NEWMAIN
   ```

   When the new MAIN is finished processing, it issues message IRRM102I:

   ```
   IRRM102I SYSTEM NEXTMAIN IS NOW THE MAIN SYSTEM IN LOCAL NODE THISMSN.
   ```

3. Update peer systems.

   Until this step is performed, a **TARGET** LIST command issued on a peer system displays the old MAIN as the main system. To update these systems, issue the **TARGET** NEWMAIN command on each system. For example:

   ```
   TARGET NODE(THISMSN) SYSNAME(NEXTMAIN) NEWMAIN
   ```

   If the peer system is lower than V2R2, use the MAIN keyword instead:

   ```
   TARGET NODE(THISMSN) SYSNAME(NEXTMAIN) MAIN
   ```

4. Update the RACF parameter library.

   Whenever a system in a multisystem node is IPLed after a dynamic main switch is performed, that system will retain no memory of the switch, and the current status will not be communicated to it automatically. To preserve a dynamic main switch across IPLs, update the parameter library to reflect the new main system.

   To harden the change in the RACF parameter library of the switching multisystem node, move the MAIN keyword from the TARGET statement of the original MAIN of the local node to that of the new MAIN. Do not put a TARGET PLEXNEWMAIN or TARGET NEWMAIN command in the parameter library.

Common errors performing a dynamic main switch

If you encounter an error while performing a dynamic main switch, error messages will guide you in correcting the error. Listed here are a few common errors you should be aware of.

1. If you issue a **TARGET** PLEXNEWMAIN or **TARGET** NEWMAIN command before having enabled **SET** FULLRRSFCOMM, and there are remote multisystem nodes, you will receive message IRRM042I on the new main system. For example:

   ```
   IRRM042I RRSF NODE THATMSN SYSNAME THATSYS2 MUST BE IN THE OPERATIVE OR DORMANT
            STATE FROM THE PERSPECTIVE OF SYSTEM NEXTMAIN TO CONFIGURE A NEW MAIN.
   ```

In this case, you must first perform the setup documented in “Allocating workspace files for remote non-MAIN connections in preparation for dynamic main switches” on page 157, on the system from whose perspective the message is issued (it can be different from the issuing system only if PLEXNEWMAIN is used). This must also be performed on the remote non-MAIN in order for it to gain the benefit of the dynamic main switch. However, performing it only on the switching system will allow the switch to continue, and to be recognized by any systems which are capable of recognizing it.

2. If the user issues **TARGET** NEWMAIN on the intended new MAIN before issuing it on the old MAIN, then he will see a series of handshaking error messages when the new MAIN attempts to re-establish connections with the systems it had just made DORMANT:

   ```
   IRRI016I ERROR: LOCAL NODE THISMSN SYSNAME NEXTMAIN AND PARTNER
   NODE THATMSN SYSNAME THATSYS1 HAVE CONFLICTING TARGET STATEMENTS WITH LOCAL SYSTEM.
   REASON CODE 4.
   ```

   Note: An APPC connection would issue the similar message IRRI014I instead.

   RRSF will protect itself against any system asserting that it is MAIN when an OPERATIVE-ACTIVE connection already exists to a MAIN system in that multisystem node. The new MAIN will still consider itself MAIN, but none of its remote connections will. To fix the mistake, issue the command again specifying the current MAIN as the MAIN system. In the case of APPC, this will result in:

   ```
   IRRI015I ERROR: HANDSHAKING HAS ALREADY FAILED WITH PARTNER NODE THATMSN SYSNAME THATSYS1.
   ```

   and a **RESTART** CONNECTION command must be issued in addition.

3. If the user changes his mind after having issued **TARGET** NEWMAIN on the old main system, but before having issued the command on the new mainsystem, the first part of the dynamic main switch can be reversed. Simply issue **TARGET** NEWMAIN on the old main, indicating its system name in the SYSNAME keyword, and it will become the MAIN again.

   When using PLEXNEWMAIN, simply switch back after the first switch is complete.

4. If the user issues **TARGET** PLEXNEWMAIN or **TARGET** NEWMAIN on the current MAIN system, specifying that same system in the SYSNAME keyword, this is not considered an error, and switch processing is performed again. That is, connections will be made DORMANT, INMSG files will be drained, and then the system will reactivate its connections as MAIN.

5. If you specify the incorrect new main system name on a non-switching peer when using NEWMAIN, then that system's MAIN bit will get turned on, and a TARGET LIST issued from this system will give incorrect output. Further complications will arise in a sysplex environment if the system in error happens to be the one queried by an IPLing peer system to obtain the name of the current MAIN system. To fix the problem, simply re-issue the **TARGET** NEWMAIN command with the correct system name.

   When using PLEXNEWMAIN, you will have initiated the switch. If it succeeds, you will need another PLEXNEWMAIN to switch the main back again.

6. If you shut down the old MAIN prior to attempting a dynamic MAIN switch, the PLEXNEWMAIN keyword will fail. You can use the NEWMAIN keyword to proceed, however there is a risk that work will run out of order. The best practice is to perform the switch prior to shutting down the old MAIN so it can drain its work in progress. Or, you should wait until the old MAIN is re-IPLed and initiate the switch then. However, if you wish to proceed, the following information might be helpful.

   When the **STOP** command is issued to shut down the RACF address space, there is an intentional built-in delay to allow work in progress to complete. So unless you have an extremely busy workload on the old MAIN, it will most likely be able to drain its INMSG completely, even if you had not initiated the switch prior to shutting down. This cannot be guaranteed.

   A **TARGET** LIST issued from one system will not display file information for a peer system, so it cannot be used to confirm the state of the down system's INMSG files. Instead, you could run the IRRBRW00 utility to look in the old main's INMSG files to see what is there, assuming they are on shared DASD. This could get tedious with a large number of remote systems, and thus INMSG files, that you would need to inspect. And, you need to know the name of every file to input to IRRBRW00. **TARGET** LIST from a peer system won't display this information, but if you have a consistent naming convention, you

can figure out the names, perhaps while browsing the RACF parameter library member to see the **TARGET** commands.

Once you are browsing the INMSG files, it can still be difficult to determine your risk of running work out of order. If the files are empty, or contain nothing but returned output, then it is safe to proceed. However, if update requests exist (for example, RACF commands, propagated passwords, etc), then it depends on whether subsequent requests are dependent upon them. For example, if an **ADDUSER** command is contained in an INMSG file, then any **ALTUSER** command for that user that is issued on the new MAIN system will fail, and RACF databases will be out of sync.

7. Part of the processing for a dynamic main switch is to allow the old MAIN to drain its work in progress so that work will not be run out of order during the switch. If the draining process takes longer than 30 minutes, the NEWMAIN or PLEXNEWMAIN command will fail with error message IRRM100I.

   For example, if you are using PLEXNEWMAIN to change the main from OLDMAIN to NEXTMAIN in node THISMSN, and because there is so much work in progress on system OLDMAIN, the command times out, you might see the following:

   On system PEER1 you issue PLEXNEWMAIN:

   ```
   TARGET PLEXNEWMAIN NODE(THISMSN) SYSNAME(NEXTMAIN)
   IRRM002I RSAJ SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
   ```

   ...
   ...30 minutes later...
   ...

   The following is displayed on system OLDMAIN:

   ```
   IRRM100I WHILE PROCESSING A DYNAMIC MAIN SWITCH, RRSF COULD NOT
   CONFIRM THAT ALL INMSG FILES ARE EMPTY ON THE OLD MAIN SYSTEM OLDMAIN.
   ```

   The following is displayed on system PEER1 (where the PLEXNEWMAIN command was issued):

   ```
   IRRM112I RSAJ SUBSYSTEM TARGET COMMAND ENCOUNTERED AN ERROR ON THE OLD MAIN
   SYSTEM OLDMAIN. USE THE TARGET NEWMAIN COMMAND TO COMPLETE THE MAIN SWITCH.
   ```

   Or if you are using NEWMAIN to change the main from OLDMAIN to NEXTMAIN in node THISMSN, you might see the following:

   ```
   TARGET NEWMAIN NODE(THISMSN) SYSNAME(NEXTMAIN)
   IRRM002I RSAJ SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
   IRRM098I DRAINING SYSTEM OF INBOUND WORK. DO NOT INITIATE THE MAIN SWITCH ON THE
   NEW MAIN SYSTEM UNTIL MESSAGE IRRM099I IS ISSUED.
   ```

   ...
   ...30 minutes later...
   ...

   ```
   IRRM100I WHILE PROCESSING A DYNAMIC MAIN SWITCH, RRSF COULD NOT CONFIRM THAT
   ALL INMSG FILES ARE EMPTY ON THE OLD MAIN SYSTEM OLDMAIN.
   ```

   Once the dynamic main switch fails in this way, there is no system acting as main for the local node. See the description of message IRRM100I in "RACF Messages and Codes" for a full description. The best solution is to allow the INMSG files to continue draining until they are empty, and then go back to the original state by issuing a TARGET NEWMAIN command on the old MAIN system to make it main again (even if the switch was initiated using the PLEXNEWMAIN operand):

   For example, on system OLDMAIN issue:

   ```
   TARGET NEWMAIN NODE(THISMSN) SYSNAME(OLDMAIN)
   ```

   Then you should perform the dynamic main switch at a time when the system is not so overloaded with RRSF work.

Changing the protocol for a connection

You can change the protocol that a connection uses. For example, if two nodes are communicating using APPC/MVS, you can change the protocol the nodes use to TCP/IP. The procedure shown here for changing the protocol for a connection has the following characteristics:

- The original protocol continues to work until the new protocol successfully establishes a communications channel.
- No updates are lost during the conversion. Any requests that are queued in the workspace data sets for the original protocol are processed.
- The order of the requests is maintained.
- The conversion process can be restarted after an abend or the stop and restart of the subsystem address space without losing updates.
- The conversion process is bidirectional; it can convert an APPC/MVS connection to TCP/IP, or a TCP/IP connection to APPC/MVS.
- During the conversion, there will be two sets of workspace data sets for the connection, one for the old protocol and one for the new protocol. Both sets will be shown if you issue a TARGET LIST command during the conversion.
- The original node definition and its workspace data sets are deleted when the conversion process completes.

Steps for changing the protocol for a connection

About this task

Before you begin:

- Ensure that you have done all setup that is required for the new protocol that the connection will use.
  - For the setup required for TCP/IP and AT-TLS, see "Setting up your system to use TCP/IP" on page 153.
  - For the setup required for APPC/MVS and VTAM, see "Setting up your system to use APPC/MVS and VTAM " on page 149.
- Ensure that you have sufficient DASD space available for the two sets of workspace data sets that exist temporarily during the conversion process.
- If you are changing the protocol in a network that has multisystem nodes with a shared parameter library, review "Sharing a RACF parameter library on a multisystem node" on page 190.

Guideline: Perform the conversion when the workspace data sets are as close to empty as possible, and are not receiving large amounts of updates. Doing this allows a quicker conversion. The procedure handles many queued requests in the old workspace data sets, but many queued requests lengthens the conversion process and increases the risk of a disruption.

Perform the following steps to convert the protocol for a connection from APPC to TCP/IP or from TCP/IP to APPC.

Procedure

1. On the local node, create a new RACF parameter library member to contain the TARGET command for the remote node using the new protocol.

   _____

2. At the end of your current parameter library member, add a SET INCLUDE(*xx*) command where *xx* is the suffix of the new member.

   For a multisystem node that shares a parameter library, the new parameter library member are shared by all systems in the multisystem node.

   _____

3. On the local node, issue a TARGET command from the console to make the local node dormant because you can only change protocol information about a node that is dormant. Then issue a TARGET command for the local node specifying the new protocol, and making it operative.

   This command causes a listener process for the new protocol to be established on the local node. For example, if the local node is named NODE1 and you are converting the protocol from APPC to TCP/IP:

   ```
   TARGET NODE(NODE1) DORMANT
   TARGET NODE(NODE1) PROTOCOL(TCP) OPERATIVE
   ```

   You should see the following message on the console:

   ```
   IRRC054I RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY ESTABLISHED.
   ```

   Inspect the messages that are issued and debug any problems that occur.

   For a multisystem node whose member systems share a parameter library, perform this step on each system in the multisystem node.

_____

4. On the local node, update the existing parameter library member:
   a) Add the TARGET command specifying the new protocol that you issued in step "3" on page 182 immediately after the existing TARGET command for the local node.
   b) Remove the OPERATIVE keyword from the existing TARGET command for the local node. Only the final TARGET command for the local node should specify OPERATIVE.
5. Repeat steps "1" on page 181 to "4" on page 182 on the remote system.

_____

6. On both systems, copy the existing TARGET command for the remote node into the new parameter library member.

   Edit the TARGET command:

   - Replace the old protocol information with the new protocol information.
   - Adjust the workspace data set names if you want to. It is a good idea to keep the same PREFIX value. If the new workspace data sets are not protected by the RACF DATASET profiles protecting the existing workspace data sets, make sure that the new workspace data sets are protected.

   For a multisystem node whose systems share a parameter library, if any systems on the multisystem node are not running z/OS V1R13 or later, commands that specify TCP/IP fail. That is expected and is not a problem. The systems continue communicating using APPC.

   Tip: If the console command line is not long enough to specify the complete TARGET command, use keyword abbreviations, or break the TARGET command into multiple commands.

   Note: After the completion of the conversion, edit the parmlib to expand the keyword abbreviations.

_____

7. On both systems, issue the new TARGET command from the console, and debug any problems that occur.

   These TARGET commands cause RACF to begin the conversion process. If you make any changes to a TARGET command while debugging, make the same changes to the TARGET command in the parameter library. Expect the first TARGET command to fail because a protocol mismatch is in effect until the new TARGET commands are issued on *both* systems. For example, you might receive this message:

   ```
   IRRI016I ERROR: LOCAL NODE NODE1 AND PARTNER NODE NODE2 HAVE CONFLICTING TARGET
   STATEMENTS WITH LOCAL SYSTEM. REASON CODE 5.
   ```

   After you issue the new TARGET command on the second system, the protocol mismatch is resolved, and the conversion process continue. For example, you might receive this message:

```
IRRC057I RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE NODE2 HAS BEEN
INITIATED.
```

RACF establishes a connection using the new protocol. There might be a message on the console indicating that the connection has been established, similar to:

```
IRRI027I RACF COMMUNICATION WITH TCP NODE NODE2 HAS BEEN SUCCESSFULLY
ESTABLISHED USING CIPHER ALGORITHM 35 TLS_RSA_WITH_AES_256_CBC_SHA.
```

Tip: If the console command line is not long enough to specify the complete TARGET command, use keyword abbreviations, or break the TARGET command into multiple commands.

During the conversion, there are two sets of workspace data sets for the connection, one for the old protocol and one for the new protocol. Both sets are shown if you issue a TARGET LIST command during the conversion. The set for the old protocol is displayed under the heading "CONVERSION FILE". For an example see Figure 25 on page 183.

```
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
   STATE - OPERATIVE ACTIVE
   .
   .
   .
   WORKSPACE FILE SPECIFICATION
         PREFIX                    - "SYS1.RRSF"
         WDSQUAL                   - <NOT SPECIFIED>
         FILESIZE                  - 500
         VOLUME                    - TEMP01
         FILE USAGE
                 "SYS1.RRSF.NODE1.NODE2.INMSG"
                                   - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
                 "SYS1.RRSF.NODE1.NODE2.OUTMSG"
                                   - CONTAINS 0 RECORD(S)
                                   - OCCUPIES 1 EXTENT(S)
         CONVERSION FILE
                 INMSG WORKSPACE FILE NOT ALLOCATED
                 "SYS1.RRSF.MF1AP001.MF2AP001.OUTMSG"
                                  - CONTAINS 5 RECORD(S)
                                  - OCCUPIES 1 EXTENT(S)
```

Figure 25. TARGET LIST output showing workspace data sets for both the old and new protocols during a protocol conversion

8. Wait for the conversion to complete successfully.

   When the conversion is complete, all requests that were queued in the old workspace data sets are complete, and you receive message IRRC058I on the consoles of both systems. It is similar to:

```
IRRC058I RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE NODE2 IS COMPLETE.
```

   Be sure that you wait until you have seen message IRRC058I on *both* systems. Then delete the TARGET command for the old protocol from the original parameter library member on each system.

   Note:

   a. Leaving the TARGET commands for the old protocol in the parameter library members until the conversion completes ensures that if the subsystem is stopped before the conversion completes, it can resume on the next IPL without losing any of the queued requests in the old workspace data sets.

   b. For a multisystem node whose member systems share a parameter library, do not delete the TARGET command for the old protocol if any member system still needs to use the old protocol. On subsequent IPLs, this command continues to be issued for the systems that use the new protocol. The old protocol is temporarily defined, and then an automatic conversion to the new protocol occurs, and you receive messages indicating that the conversion is starting and that the conversion is complete. Be sure that you have enough DASD space for the workspace files for the old protocol, which is briefly allocated and deleted. If you want to avoid this behavior, split your shared parameter library into two members: one for the systems that use the old protocol, containing only

TARGET commands for the old protocol, and a corresponding member for use by systems that use the new protocol.

Results

When you are done, you have changed the protocol used for a connection. All requests that were queued in the workspace data sets for the original protocol have completed.

If you want to change the protocol for connections to other remote nodes, repeat steps “6” on page 182 to “8” on page 183 for each of those connections. If you change the protocol for all the remote nodes, and you no longer intend to use the old protocol, when the protocol conversions are complete you can remove the old protocol information from the TARGET command for the local node in the parameter library member. If the old protocol was APPC/MVS, after you remove the old APPC protocol information from the TARGET command for the local node, the APPC server will no longer be registered when the address space restarts. If you do not want to wait for the next IPL to shut down the APPC server, you can shut it down immediately by following these steps:

1. Make the local node dormant:

```
TARGET NODE(local_node) DORMANT
```

2. Delete the APPC protocol instance from the local node:

```
TARGET NODE(local_node) PROTOCOL(APPC) DELETE
```

3. Make the local node operative:

```
TARGET NODE(local_node) OPERATIVE
```

When you have converted the protocol for all intended nodes, you can optionally combine the two parameter library members (the original one and the one that you created in step “1” on page 181) into one by moving the statements for the new protocol into the original member and deleting the SET INCLUDE command you added to the original member in step “1” on page 181.

Changing the protocol for the entire network

To change the protocol for the entire network, follow the instructions in “Changing the protocol for a connection” on page 181 for each connection in the network. After you are done, you should consider what you want to do with the old configuration information, including TARGET statements, RACF profiles in classes such as APPCLU, APPCSERV, and VTAMAPPL (if the old protocol was APPC/MVS) or SERVAUTH (if the old protocol was TCP/IP), and VTAM configuration or TCP/IP configuration. You might choose to keep the old configuration information, so that you can switch to the old protocol if you have problems with the new protocol. Or you might choose to delete it all.

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 by way of a DATASET profile. If you have not defined the RACF subsystem as trusted or privileged, make sure that it has READ access to the RACF parameter library.

Note:

1. 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.
2. An operator can issue a SET INCLUDE command to process a member that contains commands that the operator does not have authority to issue directly.

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 suggested.

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

Note:

You must ensure the user under which the RACF subsystem runs has sufficient authority to execute the desired commands. See the *z/OS Security Server RACF Command Language Reference* for the authorization requirements of individual commands.

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.

Note: The RACF commands allow you to abbreviate an operand to the least number of characters that uniquely identify the operand. To avoid conflicts in abbreviations, it is a good practice to fully spell out all operands on commands that are hardcoded in 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.

Blank lines and comments in the RACF parameter library

Blank lines and whole-line-comments are allowed. A whole-line-comment is any line that starts with //, in any column. Specifying a continuation character in a whole-line-comment has no effect; it is treated as part of the comment. A whole-line-comment or blank line may not be specified within a continued command.

Note:

The minus (-) sign is the only continuation character that is supported.

The following are examples of valid whole-line-comments:

- `//This is a comment line`
- `// This is a comment line //`
- `                                    // This is a comment line`

The following are examples of incorrect whole-line-comments:

```
// This is a comment, trailing dash ignored  -
This is treated as a new command, not a continuation of above comment.

TARGET LIST   // This is not a valid comment and will fail

TARGET   -
// This will be treated as part of a TSO command and will be failed when it runs
   LISTPROTOCOL
```

When adding comments to a parameter library that is shared among systems of a multisystem node, and some of them are earlier than V2R1, message IRRC003I is issued on those systems for every blank line and whole comment line.

TSO parsing services are used to process the actual commands that are contained within the parameter library. Therefore, you can also use delimiter /* and */ to create comments within a command image, according to the rules enforced by TSO. For more information about using comments in TSO commands, see TSO/E commands and subcommands in *z/OS TSO/E Command Reference*.

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.

Restriction: You cannot concatenate data sets under the RACFPARM DD name. If your JCL specifies concatenated data sets for RACFPARM, they are ignored.

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 185 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 26 on page 188 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.

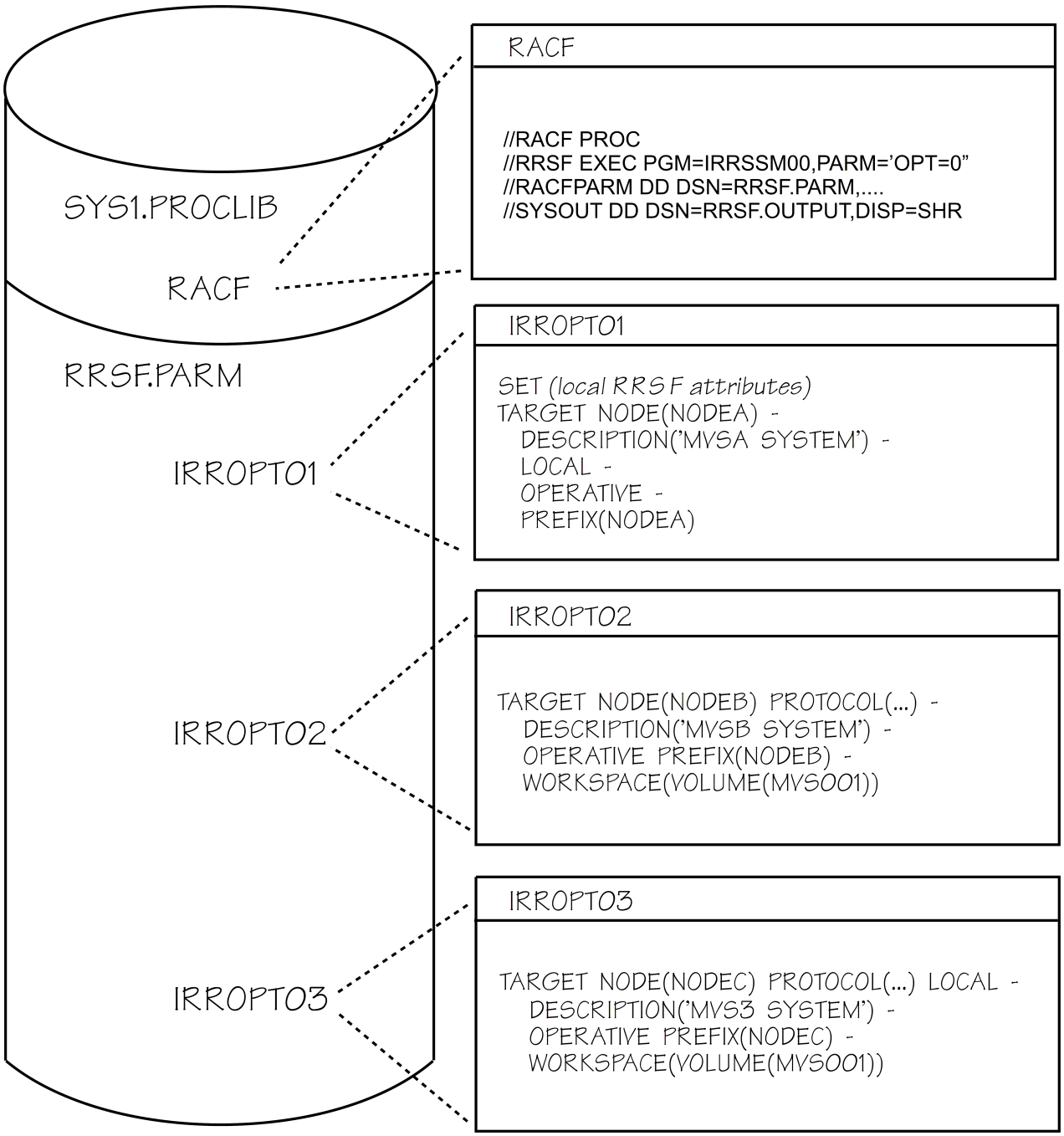


Figure 26. Example of a RACF parameter library for a node running in local mode

If you do not want a parameter library member processed automatically

You might want to use a RACF parameter library, but not have a member that is 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, do not 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.

If you are using the RACF parameter library that is shown in Figure 26 on page 188, 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 27 on page 190.

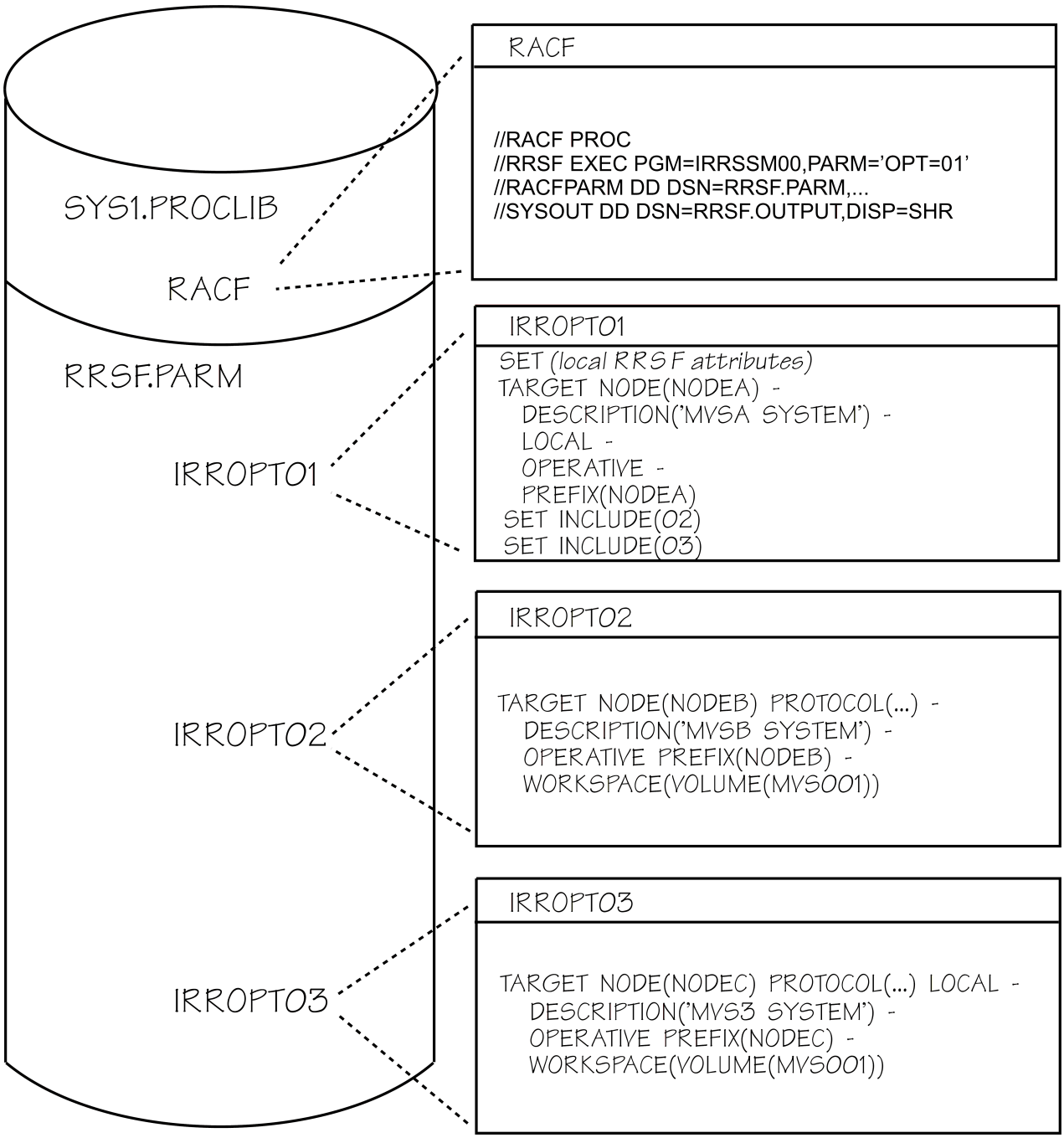


Figure 27. Example of a RACF parameter library for a node running in remote mode

A parameter library member that is included in another one by way of 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 27 on page 190 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 by way of RRSF, each member system of a multisystem node 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 should 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 196, a multisystem node named NODEAB has two member systems, MVSA and MVSB. MVSA is the main system. This node communicates by way of 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.

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 329 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 193 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 on these remote sharing functions and off as you activate and deactivate the class.

Guideline: RACLIST the RRSFDATA class.

"Customizing a remote sharing environment" on page 191 shows the RRSFDATA resource names and the remote sharing functions that they control.

Table 16. RRSFDATA resource names. The node name on a resource name is the name defined for a node by the TARGET command. For more information about defining node names, See "Configuring an RRSF network" on page 154.

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.
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.PHRSSYNC	Have RACF automatically direct all password phrase changes 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*.
IRR.RRSF.CONNECT	Connect to the local node when the AT-TLS rule covering the connection specifies a client authentication level of SAFCheck. (TCP/IP protocol only).
IRRBRW00	Execute the workspace data set VSAM file browser, IRRBRW00.

Table 16. RRSFDATA resource names. The node name on a resource name is the name defined for a node by the TARGET command. For more information about defining node names, See "Configuring an RRSF network" on page 154. *(continued)*	
Resource Name	**Controls Authorization To ...**
PWSYNC	Synchronize passwords with another user ID after establishing an association with that user ID that specifies password synchronization.
PHRASESYNC	Synchronize password phrases 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 and password phrases 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 193.)

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 do not 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 are not propagated to NODEB.

Security checks based on the RRSFDATA class are performed only on the local node, not on the remote nodes. Therefore, 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 about 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 191), 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*.

RRSF considerations for JES security

A batch job can invoke RRSF functions. For example, RACF TSO commands that are subject to automatic direction of commands can be issued from within a batch job. If your JES security approach utilizes the RACFVARS class profile &RACLNDE, it is important that all JES nodes (not RRSF nodes) that you want to be treated as local nodes are defined as members in the &RACLNDE profile. Even the JES node where the batch job is submitted needs to be a member of &RACLNDE, because there are no default members in this profile. If the submitting JES node is not defined to &RACLNDE, the RRSF authority check for the function invoked by the job (in this example an authority check for the RRSFDATA profile protecting the propagation of the particular command issued), might fail and as a result the command would not be propagated to remote RRSF nodes. For more information on the &RACLNDE profile, see Providing security for JES in *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 A, “RRSF initialization worksheet and scenario,” on page 343.

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 28 on page 194.

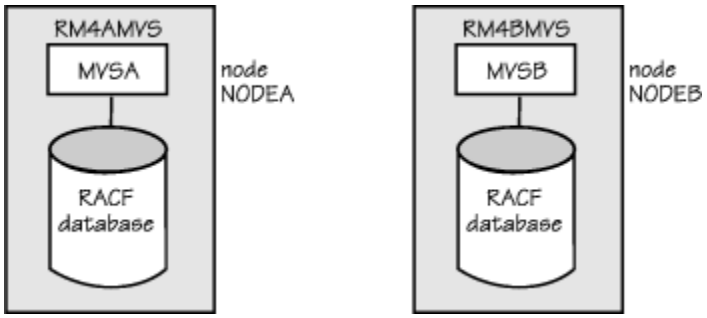


Figure 28. Two RRSF nodes in local mode

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 need the PROTOCOL keyword on the TARGET commands.

Configuring a two-node network that uses APPC/MVS

Figure 29 on page 195 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.

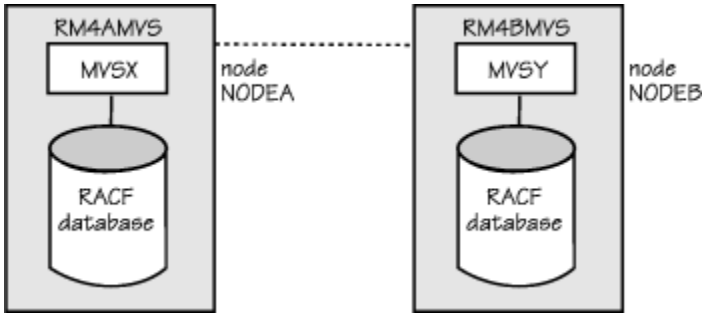


Figure 29. An RRSF network with two nodes

The following commands illustrate how you could configure this network to communicate using APPC/MVS. 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))
```

Configuring a two-node network that uses TCP/IP

The following commands illustrate how you could configure the network shown in Figure 29 on page 195 to communicate using TCP/IP. On NODEA the following command describes the local node NODEA:

```
TARGET NODE(NODEA)
       LOCAL
       DESCRIPTION('First sample node')
       PROTOCOL(TCP)
       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(TCP(ADDRESS(mvsy.example.com)
       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(TCP)
       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(TCP(ADDRESS(mvsx.example.com))
       OPERATIVE
       PREFIX(SYS1)
       WORKSPACE(VOLUME(MVS001))
```

Configuring a multisystem node

The following scenario illustrates how you could configure the RRSF network shown in Figure 30 on page 197 .

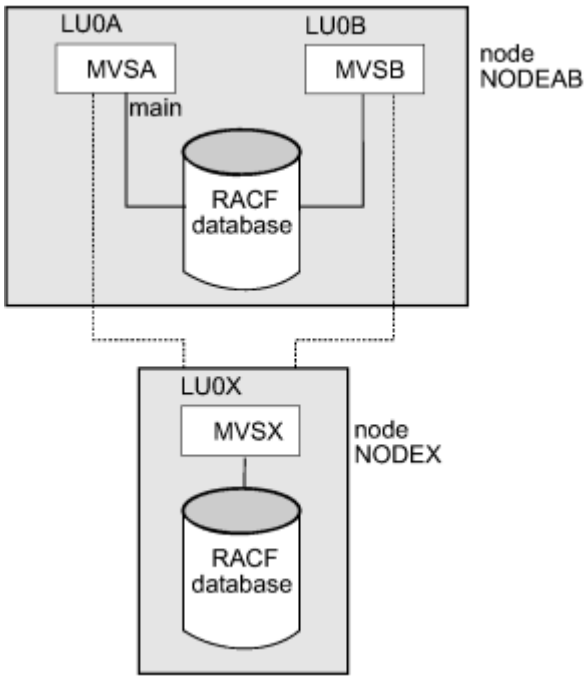


Figure 30. An RRSF network containing a multisystem node and a single-system node

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 Lube 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 A, “RRSF initialization worksheet and scenario,” on page 343.

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 147.

   _____

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 184 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 184 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.

When you are done you have configured the RRSF network shown in Figure 30 on page 197.

Configuring two multisystem nodes

The following scenario illustrates how you could configure the RRSF network shown in Figure 31 on page 198 .

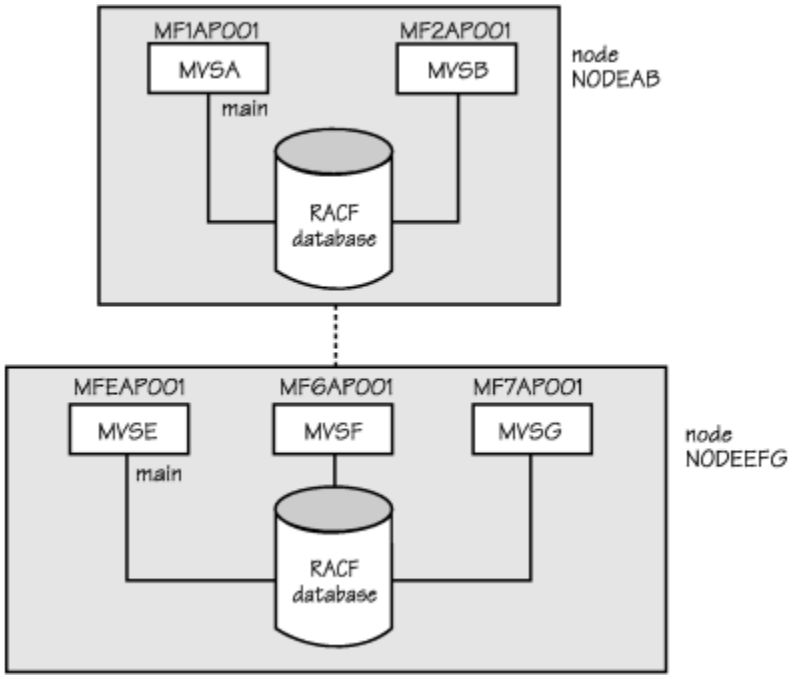


Figure 31. An RRSF network containing two multisystem nodes

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 184 for information on how to do this.)
2. If you plan to use the dynamic MAIN switching capability at any point in the future, then add the following command to IRROPTAB:

   ```
   SET FULLRRSFCOMM
   ```

   This causes the non-MAIN systems of node NODEAB to communicate with the non-MAIN systems of node NODEEFG, thereby allowing any system to become the MAIN system at any time.
3. 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
   ```
4. 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))
   ```
5. 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 184 for information on how to do this.)
6. If you plan to use the dynamic MAIN switching capability at any point in the future, then add the following command to IRROPTEG:

   ```
   SET FULLRRSFCOMM
   ```

   This will cause the non-MAIN systems of node NODEEFG to communicate with the non-MAIN systems of node NODEAB, thereby allowing any system to become the MAIN system at any time.
7. 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))
   ```
8. 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
   ```
9. Issue the following command on systems MVSA and MVSB, to cause the commands in IRROPTAB to run:

   ```
   SET INCLUDE(AB)
   ```
10. 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 APPC and TCP/IP connections between nodes

You must plan carefully to correctly implement an RRSF network. Three IBM Redbooks® can help with this planning:

- *RACF Remote Sharing Facility over TCP/IP*
- *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 composed 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 that are 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 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, see Chapter 8, “Recovery procedures,” on page 309 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 by way of 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, because 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.
5. If you use TCP/IP, periodically check the AT-TLS encryption level to ensure that has not been inadvertently weakened by a change to the AT-TLS policy. The negotiated cipher is displayed in message IRRI027I when a successful TCP/IP connection is established between two nodes, and is also displayed in the detailed TARGET LIST output for a node connected by way of TCP/IP.

Steps 1 and 2 allow for timely identification of problems you can correct before they become critical. See *z/OS Security Server RACF Diagnosis Guide* for detailed information about 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 329 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.

The IRRXUTIL interface allows gathering of real-time RRSF data from a Rexx exec. See *z/OS Security Server RACF Macros and Interfaces* for details.

Chapter 6. RACF database utilities

The RACF utilities are used for maintaining, modifying, copying, unloading, and monitoring the RACF database.

Table 17. 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 204
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 209
IRRUT100	Lists all the occurrences of a user ID or group name in the RACF database	"RACF cross reference utility program (IRRUT100)" on page 213
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 233
Note: For a summary of RACF utilities described in other areas of the RACF library, see "Utilities documented in other documents" on page 244.		

Note:

1. If you are sharing a database between z/OS and z/VM, run the utilities from the z/OS side for ease of recovery and error reporting.
2. If you are sharing a database, the templates must match the latest level of code on the sharing systems. Run the IRRMIN00 utility for the latest release to update the database templates.
3. Run z/OS Security Server (RACF) utilities only on a z/OS Security Server (RACF) system. Do not use RACF utilities with an earlier release of RACF, and do not run utilities from an earlier release of RACF on your system. The exceptions to this are IRRMIN00 and IRRUT100, which can be run on a lower-level system.
4. In general, if you are sharing a RACF database between systems at different levels, you can run any of the utilities, except IRRMIN00 and IRRUT400, from any of the sharing systems. For example, if a z/OS V1R5 system is sharing a database with a z/OS V1R6 system, you can run the IRRUT200 utility from either the V1R5 system or the V1R6 system. To get the most functionality, though, run the utility from the latest level system sharing the database. For IRRMIN00 and IRRUT400, always run the latest level of the utility. You can run IRRMIN00 on either the latest level system sharing the database, or on an

© Copyright IBM Corp. 1994, 2020

earlier system using JCL that includes a STEPLIB to an APF-authorized library that contains the latest version of the utility. Run IRRUT400 on the latest level system sharing the database. For restrictions involving the IRRIRA00 utility, see "RACF internal reorganization of aliases utility program (IRRIRA00)" on page 204.

Rules: If you are sharing a RACF database between a system running z/OS V1R8 (or higher) and a z/OS V1R4 system, you must follow these rules:

- Do *not* run the following utilities from the z/OS V1R4 system:
  - IRRMIN00
  - IRRUT200
  - IRRUT400
  - IRRUT300 (BLKUPD)
  - IRRDBU00
  - IRRIRA00
- Always run IRRUT400 from the highest level system.
- Run IRRMIN00 either from the highest level system, or from a lower level system using JCL that includes a STEPLIB to an APF-authorized library that contains the z/OS V1R8 (or higher) version of IRRMIN00.
- Run the other utilities from either a system running z/OS V1R8 (or higher) or run them from a z/OS V1R5, V1R6, or V1R7 system with APAR OA12443 installed.

5. A RACF database must not reside in the extended addressing area of DASD volumes. If a RACF database is allocated in the extended addressing area, RACF and its related utilities may not work correctly. To ensure that RACF databases are not allocated in the extended addressing area, the following DD statements for the following RACF utilities must not contain the keyword parameter EATTR unless its value is NO (EATTR=NO):
   - the SYSRACF DD statement for the IRRMIN00 utility
   - the SYSUT1 DD statement for the IRRUT200 utility
   - the OUTDD DD statement for the IRRUT400 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.
- Before running IRRIRA00 ensure there's space on the dataset. Run IRRUT200 to ensure the database has room for growth. If necessary run IRRUT400 to increase the amount of space on the RACF database. Options on the reorganization can be ALIGN, FREESPACE and LOCKINPUT (to get an intact database if the INPUT is a live active database). NOTE: As documented in the RACF database utilities topic of the RACF Systems Programmer's Guide, never run IRRUT200 or IRRUT400 with the OUTPUT specified as a live active database. RVARY the output copy to an inactive state first. If using LOCKINPUT on the input file, add another step to specify UNLOCKINPUT.
- Ensure enough CPU time for the conversion job. Check what initiator class the conversion job will run in and that it defaults to a sufficient CPU time. Large databases with many user IDs take more cpu minutes to run than smaller databases. Specifying too much is better than too little. RVARYING the backups inactive can reduce the required time Upon completion, the backups will need to be resynchronized with the primary. For more information, see Chapter 6, “RACF database utilities,” on page 203.
- Ensure that you run a database reorganization after STAGE1. Performing a resyncronization on primaries and backups, that were inactivated, affords you the opportunity to include a database reorganization.
- Consider performing a database reorganization to optimize the new index structure and optimally place data blocks.

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.

The size of an alias index entry is limited to 129 user IDs or group names each 8 characters in length (more than 129 if their average lengths are less than 8 characters). As a result, the number of user IDs that can share a UID, and the number of groups that can share a GID, are limited. IRRIRA00 fails if the size of an alias index entry is exceeded. If you exceed the limit, try to combine user ID functions for started tasks or daemons so that fewer user IDs share the same UID. If you exceed the limit because too many user IDs share UID(0), consider using profiles in the UNIXPRIV class to selectively assign superuser privileges, and reduce the number of user IDS with UID(0).

Restriction: If you are sharing a RACF database between a system running z/OS V1R8 (or higher) and a z/OS V1R4 system, do not run this utility from the z/OS V1R4 system. Run it from a system running z/OS V1R8 (or higher) or run it from a z/OS V1R5, V1R6, or V1R7 system with APAR OA12443 installed.

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 18. 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 • Purges VLF • Allows alias index entry locates	Maintains VLF	Identity search order: 1. VLF 2. Alias index entry locate

Note:

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.

Important:

- Before advancing the stage of your database, make a copy of the database for recovery purposes. If the utility fails, you might need to restore the database from a valid backup. Then resolve any conditions that caused the utility to fail and rerun the utility.
- This utility fails if more than 129 8-byte user IDs are assigned to the same UID, or more than 129 8-byte group names are assigned to the same GID. (The limit is higher for user IDs or group names that are less than 8 bytes.) You can run the ICETOOL utility to verify that no UIDs or GIDs are approaching this limit. For information on the ICETOOL utility, see *z/OS Security Server RACF Security Administrator's Guide*.

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

Important: Before advancing to stage 1, 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

Important:

- 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 does not use mapping profiles for UID, GID, SNAME, and UNAME associations. Commands such as ADDUSER no longer maintain the old mapping profiles. You can 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 8, “Recovery procedures,” on page 309 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. • Insufficient authority.
8	(8)	Processing error.
C	(12)	Terminating error. I/O error reading or writing the ICB.

Hex	(Decimal)	Meaning
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 or 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 • Maximum size of an alias index exceeded • SYSPRINT DCB open failed.
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 3.

If you have split your database, you must run IRRMIN00 against each data set defined in your data set name table. 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 information consists of a 7-character FMID or APAR level, followed by a space, followed by an 8-digit release level and an 8-digit APAR level. The 8-digit release level and the 8-digit APAR level are separated by a period (*rrrrrrrr.aaaaaaaa*). Each new RACF release increments the release level, and each APAR that ships templates increases the APAR level. The IRRMIN00 utility uses this level information to determine the relationship between different copies of the templates on the system. In the following SET LIST output, HRF7708 is the FMID of the RACF release, 00000020 is the 8-digit release level, and 00000010 is the 8-digit APAR level.

```
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. Note that RACF does not consider the 7-character FMID or APAR level when comparing the templates.

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 IRRMIN00 JCL must specify the real name of the data set; do not specify an alias.
- If you are sharing a database between systems at different levels, only run the latest level of IRRMIN00. For example, if a z/OS V1R8 system is sharing a database with a z/OS V1R7 system, only run the V1R8 version of IRRMIN00. You can run the utility either on the V1R8 system, or on the V1R7 system using JCL that includes a STEPLIB to an APF-authorized library that contains the V1R8 version of IRRMIN00.

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 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 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 8, "Recovery procedures," on page 309 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. Specify the real name of the data set; do not specify an alias.

Do not allocate this data set in the extended addressing area of DASD volumes. To ensure that this data set is not allocated in the extended addressing area, the SYSRACF DD statement must not contain the keyword parameter EATTR (unless its value is the default NO). Since EATTR=NO is the default, it is not necessary to include this on the DD statement.

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.

Hex	(Decimal)	Meaning
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 certain occurrences of a user ID or group name 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.

IRRUT100 provides information on the occurrences described in "Group name and user ID occurrences that IRRUT100 lists" on page 213. IRRUT100 does not list all occurrences in the RACF database.

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.

Guideline: Use the IRRDBU00 and IRRRID00 utilities to keep user ID and group information current in the RACF database.

For more information on IRRDBU00 and IRRRID00, see *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF Macros and Interfaces*.

To invoke IRRUT100, you must be a RACF-defined user and either have the SPECIAL, group-SPECIAL, AUDITOR, group-AUDITOR, or ROAUDIT 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 following occurrences of each user ID or group name specified. In the output, the letter G in parentheses follows each generic profile name. See Figure 32 on page 218 for a sample output of the printed report that IRRUT100 produces.

Group name and user ID occurrences that IRRUT100 lists

IRRUT100 provides information on the following occurrences:

For groups:

- The group name is defined as a group 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:

- The user ID is defined as a user 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*.

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 259.

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 8, "Recovery procedures," on page 309 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 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 32 on page 218 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
 RACLINK entry is present in user profile SIVLE as MVS2.USER1
 Owner of user USER2
 User entry exists
 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 32. Sample output from IRRUT100

The IRRUT100 utility sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.
20	(32)	IRRUT100 has ended because RACF is not enabled. For information on enabling RACF, see "Enabling and disabling RACF" on page 67. 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 can perform the following functions:

- Validate and report errors found in the relative byte addresses (RBAs) of each segment of all profiles.
- Validate that index entries point to the correct profile.
- Validate the data set format.
- Issue return codes to signal validation errors.
- Scan the index blocks and print information about problems with the index-block chains.
- Compare the segments of the data set that are actually in use to the segments allocated according to the BAM blocks, and print information about inconsistencies.
- Process the alias index to detect errors and display index blocks.
- Create 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.
- Create 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.
- Create 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.
- The JCL must specify the real name of the data set; do not specify an alias.

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 input data set. If you plan to copy an in-use primary data set to its corresponding in-use inactive backup data set (RVARY LIST output shows the in-use RACF data sets), you can specify PARM=ACTIVATE in your JCL to have IRRUT200 activate the in-use inactive backup copy without allowing the in-use active primary RACF database to be updated between the copy and activate operations, keeping the backup and primary data sets synchronized. Data set verification does not occur during PARM=ACTIVATE processing, so it's a good idea to have a data set verification step in your procedure, before PARM=ACTIVATE. If, instead of using PARM=ACTIVATE, you choose to issue an RVARY after the copy operation to activate the backup copy, there is a window of time between the copy operation and the RVARY command when the primary data set can be updated, causing the backup data set to become out of synchronization. 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

Installations can configure DFSORTs ICEGENER as a replacement for IEBGENER. In addition, there are DFSORT installation parameters which affect how ICEGENER operates. One of these installation parameters, OUTSEC, controls whether DFSORT uses automatic secondary allocation for output data sets that are temporary or new. If OUTSEC is enabled, then data sets which are allocated during the IRRUT200 job step may end up in multiple extents which can cause an I/O error and a S500 abend when the data set is used as an online RACF data base or as input to the other RACF utilities.

For information on IEBGENER, see *z/OS DFSMSdfp Utilities*. For information on installing ICEGENER as a replacement for IEBGENER, see *z/OS DFSORT Installation and Customization*. For information on how to prevent OUTSEC processing in the IRRUT200 job, see the DFSPARM DD statement in *z/OS DFSORT Application Programming Guide.*

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 can *not* be an active RACF data set. If you specify an active primary or backup data set on the system on which IRRUT200 is running, the utility fails. If you need to refresh an active RACF data set, use RVARY to deactivate the data set before running IRRUT200.

To prevent copying a data set over itself, the utility fails if you specify the same data set names for SYSRACF and SYSUT1.

Diagnostic capability

IRRUT200 is designed to detect errors in the internal organization of the RACF database when run with the INDEX and MAP functions. If you suspect a RACF database error, start your problem determination by running this utility requesting the INDEX FORMAT 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 8, “Recovery procedures,” on page 309 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 suggest 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 description of IRRDBU00 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 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

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. For RACF sysplex data sharing, if RACF is enabled for sysplex communication and is operating in data sharing or read-only mode, RACF uses ENQ instead of RESERVE.

There are four ways to run IRRUT200:

1. SYSUT1 is specified, SYSRACF is specified, PARM=ACTIVATE is not specified

   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.

   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. The data set that you specify for SYSUT1 cannot be the same data set as the one specified for SYSRACF, and cannot be an active data set on the system on which the utility is running.

   Note:

   a. This method serializes against the RACF data set only during the copy phase, which is much shorter than the verification phase.
   b. At no time is there a RESERVE on the work data set.

2. SYSUT1 is not specified, SYSRACF is specified

   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. SYSUT1 is specified, SYSRACF is not specified

   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.

4. SYSUT1 is specified, SYSRACF is specified, PARM=ACTIVATE is specified.

   If you specify PARM=ACTIVATE, SYSRACF is an in-use active primary RACF data set and SYSUT1 is the corresponding in-use inactive backup data set. RACF copies SYSRACF to SYSUT1 under serialization, and activates SYSUT1 before releasing the RESERVE. IRRUT200 diagnostics do *not* run, and SYSIN and SYSPRINT are ignored.

   For information regarding the ACTIVATE parameter and sysplex communications mode, see the information on the EXEC statement in “Input and output for IRRUT200” on page 222.

   Guideline: Run IRRUT200 using one of the first 3 methods before you run with PARM=ACTIVATE, to verify the primary RACF data set.

   Note: When PARM=ACTIVATE is specified with SYSUT1, the data set must be catalogued prior to running this jobstep. The data set can be created and cataloged in an IEFBR14 jobstep prior to this jobstep.

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 225) 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.

You can specify PARM=ACTIVATE on the EXEC statement. Specifying this parameter indicates the following:

- The input data set pointed to by SYSRACF is an active primary RACF data set on the system on which the utility is running.
- The output data set pointed to by SYSUT1 is the corresponding inactive backup RACF data set on the system on which the utility is running.
- The output data set should be activated after the copy completes. No activation password is required to activate the data set.

If the output data set is not the inactive backup data set, both the copy and the activation fail.

The ACTIVATE parameter can ensure that no updates are made to the input data set between the time that it is copied and the time that the copy is activated. However, it can only ensure a synchronized copy if the system on which the utility is running is in RACF sysplex communications mode, or the RACF data set is not shared with another system. If other systems share the backup data set and are not in sysplex communications mode, IRRUT200 can only activate the data set on the system on which the utility is running. To activate the backup data set on the sharing systems, you must issue an RVARY ACTIVE.

The intent of the ACTIVATE parameter is to create a synchronized copy of an active RACF data set, not perform diagnosis. Therefore, the control statements specified in SYSIN are ignored when the ACTIVATE parameter is specified. If you need to verify the RACF data set, do that before you make a copy of it.

If RACF is enabled for sysplex communication in a sysplex with multiple members, to run IRRUT200 specifying PARM=ACTIVATE from one of the members, at least one of the following must be true:

- The RACF sysplex communication group (IRRXCF00) is in data sharing mode.
- SYSZRACF RESERVEs are being converted to ENQs.

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. Specify the real name of the data set; do not specify an alias.

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. If you specify both SYSRACF and SYSUT1, SYSUT1 cannot point to an active RACF data set on this system. If you specify the same data set that you specify for SYSRACF, IRRUT200 fails.

Do not allocate this data set in the extended addressing area of DASD volumes. To ensure that this data set is not allocated in the extended addressing area, the SYSUT1 DD statement must not contain

the keyword parameter EATTR (unless its value is the default NO). Since EATTR=NO is the default, it is not necessary to include this on the DD statement.

For information on installing ICEGENER as a replacement for IEBGENER, see *z/OS DFSORT Installation and Customization*. For information on how to prevent OUTSEC processing in the IRRUT200 job, see the *z/OS DFSORT Installation and Customization*.

Note:

1. 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.
2. When PARM=ACTIVATE is specified with SYSUT1, the dataset must be catalogued prior to running this jobstep. The dataset may be created and cataloged in an IEFBR14 jobstep prior to this jobstep.

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.

SYSIN is ignored when PARM=ACTIVATE is specified on the EXEC statement.

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.

If you specify PARM=ACTIVATE on the EXEC statement, SYSPRINT is not used, and you do not need to specify it.

DFSPARM DD (Optional)
Defines a data set which contains overrides for the installation DFSORT defaults. The data set must be defined with RECFM of F or FB with a LRECL of 80 bytes. The DFSPARM DD statement is only applicable if all are true:

- Using SYSUT1 to create a copy of the RACF data base.
- The SYSUT1 data set is allocated as DISP=NEW.
- The SYSUT1 data set is going to be used as a RACF data set or as input to one of the RACF utilities.
- Your installation has installed the DFSORT ICEGENER utility as a replacement for IEBGENER.
- Your installation uses OUTSEC as a DFSORT default.

The data that is input to this statement is:

```
OPTION NOOUTSEC
```

Guideline: Do not run IRRUT200 under TSO, because doing so can increase the time that the RACF data set is RESERVEd. However, if you do 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, you should allocate it to SYSOUT, because IRRUT200 might produce a large volume of output.

IRRUT200 examples

In the following example, IRRUT200 copies a RACF data set to the SYSUT1 temporary 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
/*
```

In the following example, IRRUT200 synchronizes the primary and backup RACF data sets. Before you submit this JCL, you must issue an RVARY command to make the backup data set inactive. After running the job, the backup will be a copy of the primary, and both data sets will be active.

```
//COPYDS  JOB  ,'RACF SYNC BACKUP',
//     MSGLEVEL=(1,1),TYPRUN=HOLD
//STEP      EXEC  PGM=IRRUT200,PARM='ACTIVATE'
//SYSRACF   DD    DSN=SYS1.RACF,DISP=SHR
//SYSUT1    DD    DSN=SYS1.RACF.BACKUP,DISP=OLD
//SYSUT2    DD    SYSOUT=A
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
    INDEX
    MAP
    END
/*
```

The following example is similar to the prior example, except that the SYSUT1 data set is a non-temporary data set and may be used as an online RACF data set or as input to other RACF utilities. In this case, to ensure that the data set is allocated as an unmoveable, contiguous and single extent data set, CONTIG must be specified and the //DFSPARM DD statement has been added to allow the specification of NOOUTSEC.

Note: The data set allocated to SYSRACF and the data set being allocated by SYSUT1 must be on the same type of device (3380 or 3390).

```
//VERIFY   JOB
//STEP     EXEC PGM=IRRUT200
//SYSRACF  DD DSN=SYS1.RACF,DISP=SHR
//SYSUT1   DD UNIT=SYSALLDA, SPACE=(CYL,10,,CONTIG),
//         DSN=PROD.RACF.BACKUP,
//         DCB=(LRECL=4096,RECFM=F,DSORG=PSU)
//SYSUT2   DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//DFSPARM  DD *
    OPTION NOOUTSEC
/*
//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.

```
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.

```
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 33 on page 227 for sample output that IRRUT200 produces when you request formatted index blocks.

```
                                    ****  INDEX BLOCK VERIFICATION  ****
-                                 ****  SCAN OF INDEX BLOCKS AT LEVEL 02  ****

 BLOCK WITH RBA OF 00000001B000

 OFFSET  COMP.                        ENTRY NAME                                    RBA           BAM
         COUNT                                                                            BLOCK  BYTE BIT
  00E    0000  DIGTCERT-04                                                    00000000E000   00   030   0
  02C    0000  255 X'FF's                                                     00000001A000   00   048   0

 TOTAL NAMES IN THIS BLOCK-002. UNUSED BYTES-3773. AVERAGE NAME LENGTH-133.
 LEVEL NUMBER-02. DISPLACEMENT TO LAST KEY-002C. DISPLACEMENT TO FREE SPACE-013F
 (G) - ENTITY NAME IS GENERIC

-                                 ****  SCAN OF INDEX BLOCKS AT LEVEL 01  ****

 BLOCK WITH RBA OF 00000000E000

 OFFSET  COMP.                        ENTRY NAME                                    RBA           BAM
         COUNT                                                                            BLOCK  BYTE BIT
  00E    0000  irrcerta                                                       00000001D000   00   04E   0
  02A    0003  irrmulti                                                       00000000D300   00   02E   3
  043    0003  irrsitec                                                       00000000D100   00   02E   1
  05C    0000  ADAM                                                           000000014300   00   03C   3
  074    0000  ADRIAN                                                         000000016E00   00   041   6
  08E    0000  ALAN                                                           000000016F00   00   041   7
  0A6    0000  ALEX                                                           000000017000   00   042   0
  0BE    0000  ALICIA                                                         000000017100   00   042   1
  0D8    0000  ANN                                                            000000017200   00   042   2
  0EF    0000  BELINDA                                                        000000017300   00   042   3
  10A    0000  BETTY                                                          000000017400   00   042   4
  123    0000  BILL                                                           000000017500   00   042   5
  13B    0000  BOB                                                            000000017600   00   042   6
  152    0000  BOB.* (G)                                                      000000017E00   00   043   6
  16C    0000  BRAD                                                           000000017A00   00   043   2
  184    0000  BRENDA                                                         000000000D800   00   02F   0
  19E    0000  BRENDA.* (G)                                                   000000017D00   00   043   5
  1BB    0000  DASDVOL -D001                                                  000000017700   00   042   7
  1DC    0000  DASDVOL -D002                                                  000000017800   00   043   0
  1FD    0000  DASDVOL -D003                                                  000000017900   00   043   1
  21E    0000  DIGTCERT-0D8B4FEEAAD2185BF4756A9D29E17FFB.OU=Class¢1¢Public¢   000000011000   00   036   0
               Primary¢Certification¢Authority.O=VeriSign,¢Inc..C=US
                                   SEGMENT NAME: CERTDATA                    000000012200   00   038   2
  2AA    0000  DIGTCERT-00.personal-basic@thawte.com.CN=Thawte¢Personal¢Bas   000000011300   00   036   3
               ic¢CA.OU=Certification¢Services¢Division.O=Thawte¢Consulting
               .L=Cape¢Town.SP=Western¢Cape.C=ZA
                                   SEGMENT NAME: CERTDATA                    000000011500   00   036   5
  35E    0000  DIGTCERT-00.personal-freemail@thawte.com.CN=Thawte¢Personal¢   000000010800   00   035   0
               Freemail¢CA.OU=Certification¢Services¢Division.O=Thawte¢Cons
               ulting.L=Cape¢Town.SP=Western¢Cape.C=ZA
                                   SEGMENT NAME: CERTDATA                    000000010A00   00   035   2
  418    0000  DIGTCERT-00.personal-premium@thawte.com.CN=Thawte¢Personal¢P   000000010E00   00   035   6
               remium¢CA.OU=Certification¢Services¢Division.O=Thawte¢Consul
               ting.L=Cape¢Town.SP=Western¢Cape.C=ZA
                                   SEGMENT NAME: CERTDATA                    000000012900   00   039   1
  4D0    0000  DIGTCERT-00B92F60CC889FA17A4609B85B706C8AAF.OU=VeriSign¢Trus   000000015000   00   03E   0
               t¢Network.OU=(c)¢1998¢VeriSign,¢Inc.¢-¢For¢authorized¢use¢on
               ly.OU=Class¢2¢Public¢Primary¢Certification¢Authority¢-¢G2.O=
               VeriSign,¢Inc..C=US
```

Figure 33. Sample output of formatted index produced by IRRUT200 (part 1 of 3)

```
                                SEGMENT NAME: CERTDATA                      000000015200   00   03E   2
  5B2    0000  DIGTCERT-00ECA0A78B6E756A01CFC47CCC2F945ED7.CN=VeriSign¢Clas      000000016800   00   041   0
               s¢4¢Public¢Primary¢Certification¢Authority¢-¢G3.OU=(c)¢1999¢
               VeriSign,¢Inc.¢-¢For¢authorized¢use¢only.OU=VeriSign¢Trust¢N
               etwork.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000001CA00   00   04D   2
  69D    0000  DIGTCERT-008B5B75568454850B00CFAF3848CEB1A4.CN=VeriSign¢Clas      000000014100   00   03C   1
               s¢1¢Public¢Primary¢Certification¢Authority¢-¢G3.OU=(c)¢1999¢
               VeriSign,¢Inc.¢-¢For¢authorized¢use¢only.OU=VeriSign¢Trust¢N
               etwork.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      000000014A00   00   03D   2
  788    0000  DIGTCERT-009B7E0649A33E62B9D5EE90487129EF57.CN=VeriSign¢Clas      000000016400   00   040   4
               s¢3¢Public¢Primary¢Certification¢Authority¢-¢G3.OU=(c)¢1999¢
               VeriSign,¢Inc.¢-¢For¢authorized¢use¢only.OU=VeriSign¢Trust¢N
               etwork.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000001C000   00   04C   0
  873    0000  DIGTCERT-01.premium-server@thawte.com.CN=Thawte¢Premium¢Serv      00000000FD00   00   033   5
               er¢CA.OU=Certification¢Services¢Division.O=Thawte¢Consulting
               ¢cc.L=Cape¢Town.SP=Western¢Cape.C=ZA
                                SEGMENT NAME: CERTDATA                      00000000FF00   00   033   7
  92A    0000  DIGTCERT-01.server-certs@thawte.com.CN=Thawte¢Server¢CA.OU=C      00000000F200   00   032   2
               ertification¢Services¢Division.O=Thawte¢Consulting¢cc.L=Cape
               ¢Town.SP=Western¢Cape.C=ZA
                                SEGMENT NAME: CERTDATA                      000000010400   00   034   4
  9D7    0000  DIGTCERT-01A3.CN=GTE¢CyberTrust¢Root.O=GTE¢Corporation.C=US       000000011200   00   036   2
                                SEGMENT NAME: CERTDATA                      00000012D00   00   039   5
  A2D    0000  DIGTCERT-02AD667E4E45FE5E576F3C98195EDDC0.OU=Secure¢Server¢C      00000000F100   00   032   1
               ertification¢Authority.O=RSA¢Data¢Security,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000000F600   00   032   6
  AB9    0000  DIGTCERT-03.CN=GTE¢CyberTrust¢Root.O=GTE¢Corporation.C=US       000000012700   00   038   7
                                SEGMENT NAME: CERTDATA                      000000013000   00   03A   0
  B0D           SEQUENCE SET POINTER                                        00000001A000

 TOTAL NAMES IN THIS BLOCK-033. UNUSED BYTES-1192. AVERAGE NAME LENGTH-062.
 LEVEL NUMBER-01. DISPLACEMENT TO LAST KEY-0B0D. DISPLACEMENT TO FREE SPACE-0B16
 (G) - ENTITY NAME IS GENERIC


 BLOCK WITH RBA OF 00000001A000

 OFFSET  COMP.                          ENTRY NAME                                RBA          BAM
         COUNT                                                                  BLOCK  BYTE BIT
  00E    0000  DIGTCERT-04.CN=Autoridade¢Certificadora¢Raiz¢Brasileira.SP=D      000000013800   00   03B   0
               F.L=Brasilia.OU=Instituto¢Nacional¢de¢Tecnologia¢da¢Informac
               ao¢-¢ITI.O=ICP-Brasil.C=BR
                                SEGMENT NAME: CERTDATA                      000000013900   00   03B   1
  0BB    0009  DIGTCERT-2D1BFC4A178DA391EBE7FFF58B45BE0B.OU=Class¢2¢Public¢      00000000D200   00   02E   2
               Primary¢Certification¢Authority.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000000F300   00   032   3
  13E    0009  DIGTCERT-254B8A853842CCE358F8C5DDAE226EA4.OU=Class¢3¢Public¢      000000010300   00   034   3
               Primary¢Certification¢Authority.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      000000011D00   00   037   5
  1C1    0009  DIGTCERT-325033CF50D156F35C81AD655C4FC825.OU=Class¢1¢Public¢      00000000F000   00   032   0
               Primary¢Certification¢Authority.O=VeriSign,¢Inc..C=US
```

Figure 34. Sample output of formatted index produced by IRRUT200 (part 2 of 3)

```
                                SEGMENT NAME: CERTDATA                      00000000FA00   00   033   2
  244    0000  DIGTCERT-32888E9AD2F5EB1347F87FC4203725F8.OU=VeriSign¢Trust¢      000000016600   00   040   6
               Network.OU=(c)¢1998¢VeriSign,¢Inc.¢-¢For¢authorized¢use¢only
               .OU=Class¢4¢Public¢Primary¢Certification¢Authority¢-¢G2.O=Ve
               riSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000001C600   00   04C   6
  324    0009  DIGTCERT-3381F595.CN=Integrion¢Certification¢Authority¢Root.      00000000F900   00   033   1
               O=Integrion¢Financial¢Network.C=US
                                SEGMENT NAME: CERTDATA                      000000011900   00   037   1
  394    0009  DIGTCERT-35DEF4CF.OU=Equifax¢Secure¢Certificate¢Authority.O=      000000012800   00   039   0
               Equifax.C=US
                                SEGMENT NAME: CERTDATA                      000000013400   00   03A   4
  3EE    0009  DIGTCERT-38A02637.CN=Identrus¢Root¢Interoperability¢Certific      000000011100   00   036   1
               ate¢Authority.OU=Identrus¢Root¢Certificate¢Authority.O=Ident
               rus¢LLC
                                SEGMENT NAME: CERTDATA                      000000014400   00   03C   4
  47F    0009  DIGTCERT-4CC7EAAA983E71D39310F83D3A899192.OU=VeriSign¢Trust¢      000000013F00   00   03B   7
               Network.OU=(c)¢1998¢VeriSign,¢Inc.¢-¢For¢authorized¢use¢only
               .OU=Class¢1¢Public¢Primary¢Certification¢Authority¢-¢G2.O=Ve
               riSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      000000016A00   00   041   2
  556    0000  DIGTCERT-6170CB498C5F984529E7B0A6D9505B7A.CN=VeriSign¢Class¢      000000015600   00   03E   6
               2¢Public¢Primary¢Certification¢Authority¢-¢G3.OU=(c)¢1999¢Ve
               riSign,¢Inc.¢-¢For¢authorized¢use¢only.OU=VeriSign¢Trust¢Net
               work.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      000000015800   00   03F   0
  63F    0000  DIGTCERT-7DD9FE07CFA81EB7107967FBA78934C6.OU=VeriSign¢Trust¢      000000015E00   00   03F   6
               Network.OU=(c)¢1998¢VeriSign,¢Inc.¢-¢For¢authorized¢use¢only
               .OU=Class¢3¢Public¢Primary¢Certification¢Authority¢-¢G2.O=Ve
               riSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000016000   00   040   0
  71F    0009  DIGTCERT-70BAE41D10D92934B638CA7B03CCBABF.OU=Class¢3¢Public¢      00000000D000   00   02E   0
               Primary¢Certification¢Authority.O=VeriSign,¢Inc..C=US
                                SEGMENT NAME: CERTDATA                      00000000DD00   00   02F   5
  7A2    0000  IBMUSER                                                     00000000DB00   00   02F   3
  7BD    0000  SECLABEL-SYSHIGH                                           00000000D400   00   02E   4
  7E1    0000  SECLABEL-SYSLOW                                            00000000D500   00   02E   5
  804    0000  SECLABEL-SYSMULTI                                          00000000D700   00   02E   7
  829    0000  SECLABEL-SYSNONE                                           00000000D600   00   02E   6
  84D    0000  SYSCTLG                                                    00000000DA00   00   02F   2
  868    0000  SYS1                                                       000000017B00   00   043   3
  880    0000  VSAMDSET                                                   00000000D900   00   02F   1
  89C    0000  255 X'FF's
  9A8           SEQUENCE SET POINTER                                      000000000000

 TOTAL NAMES IN THIS BLOCK-021. UNUSED BYTES-1573. AVERAGE NAME LENGTH-093.
 LEVEL NUMBER-01. DISPLACEMENT TO LAST KEY-09A8. DISPLACEMENT TO FREE SPACE-09B1
 (G) - ENTITY NAME IS GENERIC

1                                    ****  SEQUENCE SET RBAS  ****
       RBA  00000000E000
       RBA  00000001A000
1                                   ****  INDEX FUNCTION STATISTICS  ****
0TOTAL NUMBER OF NAMES IN RACF DATA SET 00000056
 TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000003
 AVERAGE NUMBER OF NAMES PER INDEX BLOCK 018
 AVERAGE NAME LENGTH 076
 AVERAGE NUMBER OF UNUSED BYTES PER INDEX BLOCK 2179
 TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000002
```

Figure 35. Sample output of formatted index produced by IRRUT200 (part 3 of 3)

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 36 on page 229 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 36. 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 37 on page 232, 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 37 on page 232 for a sample printout of the encoded map that IRRUT200 produces with MAP ALL specified.

```
                                        ****  BAM BLOCK VERIFICATION  ****
0            ****  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 ******** ******** ******** ******** ******** ******** ******** ******** ********
 03B ******** ******** ******** ******** ******** ******** ******** ******** *******0 00000000 00000000 00000000 00000000
 048 I1111111 11111111 I2222222 22222222 ******** ******** ******** ******** ******** ******** ******** ******** *****000
 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
⋮
 542 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 54F 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 55C 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 569 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 576 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 583 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 590 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 59D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 5AA 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 //////// //////// ////////
 5B7 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 5C4 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 5D1 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 5DE //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 5EB //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 5F8 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 605 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
⋮
 F1C //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F29 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F36 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F43 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F50 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F5D //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F6A //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F77 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F84 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F91 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F9E //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FAB //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FB8 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FC5 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FD2 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FDF //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FEC //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FF9 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
                                        ****  MAP FUNCTION STATISTICS  ****
 NUMBER OF BAM BLOCKS DEFINED 001
 LAST BAM THAT DEFINES USED SPACE - RBA 00000000C000
 RACF DATA SET IS   4 PERCENT FULL.
 TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000003
 TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000002
 NUMBER OF GROUP    ENTRIES - 0000003
 NUMBER OF USER     ENTRIES - 0000016
 NUMBER OF DATASET  ENTRIES - 0000002
 NUMBER OF DASDVOL  ENTRIES - 0000003
 NUMBER OF DIGTCERT ENTRIES - 0000025
 NUMBER OF SECLABEL ENTRIES - 0000004
```

Figure 37. 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.

Hex	(Decimal)	Meaning
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)	RACF is not enabled. For information on enabling RACF, see "Enabling and disabling RACF" on page 67. 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.

  Restriction: Do not use IRRUT400 to merge or rearrange 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 are sharing a database between systems at different levels, only run IRRUT400 on the latest level system sharing the database. For example, if a z/OS V1R7 system is sharing a database with a z/OS V1R8 system, only run IRRUT400 from the V1R8 system
- Specify the real names of the data sets; do not specify aliases.

Guideline: Run IRRUT400 when your system has little RACF activity.

RACF sysplex communication: 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 that you specify in your JCL.

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, use IRRUT200 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 can *not* be an active RACF database. If you specify an active primary or backup data set on the system on which IRRUT400 is running, the utility fails. 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 you should use IRRUT200 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 33 on page 227, 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, fewer reads are required to access them, 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 8, "Recovery procedures," on page 309 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.

STEPLIB DD
Defines the location of an ICHRRNG-style module, which is named by the TABLE keyword, if it does not reside in the system LINKLIB. If used, the data set that is referenced by STEPLIB must be APF-authorized.

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.

Do not allocate this database in the extended addressing area of DASD volumes. To ensure that this database is not allocated in the extended addressing area, the OUTDD*n* DD statement must not

contain the keyword parameter EATTR (unless its value is the default NO). Since EATTR=NO is the default, it is not necessary to include this on the DD statement.

RANGEDD DD
Defines a partitioned data set, or a concatenation of partitioned data sets, in which to locate the member that is specified by the RANGE parameter. If RANGEDD is not defined, IRRUT400 searches the current system parmlib instead. If the RANGE parameter is not specified, RANGEDD is ignored.

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 or RANGE keyword to make the determination). See "The database range table" on page 39 and "Selecting the output data set" on page 238. If any of the input data sets are RACF-protected, you must have at least UPDATE authority for those data sets.

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). Specify the real names of the data sets; do not specify aliases.

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 1. 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.

IRRUT400 fails if an output data set is an active primary or backup data set on the system on which the utility is running, or if you attempt to copy a data set into itself by pointing to the same data set with INDD and OUTDD.

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 range definitions that indicate the range of profiles to be placed on each data set.

You can specify the name of a partitioned data set member containing the RACF parmlib range definitions with the RANGE parameter in the EXEC statement. For more information, see "Specifying parameters" on page 238. For information about the correct member syntax, see "Syntax rules for IRRPRMxx" on page 42. IRRUT400 does not use any of the data set names that are defined in the parmlib member. You must ensure that the range definitions correspond correctly to the output data sets that are specified by OUTDD*n* in the JCL.

Alternatively, you can specify the name of a module that contains an ICHRRNG-style range table in the TABLE parameter in the EXEC statement. See "Specifying parameters" on page 238. "The database range table" on page 39 describes the format of the ICHRRNG range table.

Before restarting your system to use a new split RACF database, ensure that the data set name table is consistent with the range table that is to be used for that system. The tables can be specified using a RACF parmlib member or using the ICHRDSNT and ICHRRNG load modules. Do not rely on IRRUT400 to validate all of the definitions in a RACF parmlib member. The RACPRMCK command can be used to validate the RACF parmlib member.

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" on page 238.) 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 might be allowed, or it might 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.

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 might need to use UNLOCKINPUT if you mistakenly lock the wrong data set, or if you change your mind after locking a data set.

RANGE(member)/TABLE(table-name)/NOTABLE

Use the RANGE or TABLE keyword to indicate the name ranges for separating profiles into multiple output data sets. NOTABLE is the default and implies that all profiles are to be copied into a single output data set, which is identified by OUTDD1.

The RANGE keyword enables you to specify the name of a RACF parmlib member that defines the ranges for selecting an output data set for each profile. IRRUT400 looks for the specified member in the current system parmlib concatenation unless RANGEDD is included in your JCL to indicate a different partitioned data set or concatenation. For more information about the correct member syntax, see “Syntax rules for IRRPRMxx” on page 42. Syntax errors that are found in the specified member cause IRRUT400 to terminate, even if the errors reside outside of the RANGETABLE section. The use of system symbols in the range table is not recommended, since symbol values for the environment in which you run IRRUT400 can differ from those values in another system configuration.

The TABLE keyword enables you to specify the name of an ICHRRNG-style module that defines ranges for selecting an output data set for each profile. While the TABLE value can be any member name, the member format must be that of an ICHRRNG module as described in “The database range table” on page 39. IRRUT400 loads the specified module from the system LINKLIB. You can override the location by defining a STEPLIB in your JCL. The data set that is referenced by STEPLIB must be APF-authorized.

If you are using the utility to merge or split a database into multiple output data sets, you must specify either RANGE or TABLE to indicate in which data set to place the profiles. You must ensure that the range definitions correspond correctly to the output data sets that are specified by OUTDD*n* in the JCL. IRRUT400 does not use the data set name table that is defined in a RACF parmlib member or an ICHRDSNT module. It is the system programmer's responsibility to ensure that the set of data sets that are specified by OUTDD*n* is consistent with the data set name table that you intend to use for your RACF database.

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.
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. • An INDD statement and an OUTDD statement point to the same data set • An OUTDD statement specifies an active RACF data set on this system

Hex	(Decimal)	Meaning
20	(32)	RACF 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 67. 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 might 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 can 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 2a. Splitting a database using the RANGE keyword

In this example, IRRUT400 splits a RACF database containing a single data set into two output data sets. IRRUT400 assigns profiles to the output data sets using range table definitions in a member named IRRPRM02 that does not reside in the current parmlib concatenation. The member can be found in INSTALL.WORK.PARMLIB as indicated by RANGEDD.

```
//J2      JOB
//        EXEC  PGM=IRRUT400,PARM='NOLOCKINPUT,RANGE(IRRPRM02)'
//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)
//RANGEDD   DD  DSN=INSTALL.WORK.PARMLIB,DISP=SHR
```

Example 2b. 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 an ICHRRNG-style range table in a module named RNGNEW. The load module resides in library INSTALL.LINKLIB, which is APF-authorized.

```
//J2      JOB
//        EXEC  PGM=IRRUT400,PARM='NOLOCKINPUT,TABLE(RNGNEW)'
//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 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 7. RACF installation exits

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 that are offered by RACF, and also 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 and IRRVAF01, 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 and IRRVAF01, you must re-IPL MVS for the changes to take effect. IRREVX01 and IRRVAF01 are defined to the dynamic exits facility, and you can update them without re-IPLing.

RACF initialization message ICH508I displays the names of the exits that are active for the IPL. Because IRREVX01 and IRRVAF01 are defined to the MVS dynamic exits facility, if either of their names appear in the ICH508I message, at least one active routine is added to the indicated exit point at this particular time in the IPL.

The exit routines must be reenterable and refreshable and must be 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 about these considerations, see “Installation exit considerations” on page 147.

© Copyright IBM Corp. 1994, 2020

You should provide error recovery for your exits to handle an abend situation and either recover from the situation, or, if recovery is not possible, clean up system resources such as locks and storage obtained by the exit. For information about coding error recovery procedures, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 and IRRVAF01, 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 or IRRVAF01 when at least one active exit routine is defined for the exit point at the time the report is created. The report does not include the routine names or sizes, and lists the length of IRREVX01 and IRRVAF01 as "NA" (not available).

Note: You must have the AUDITOR or the ROAUDIT 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 are 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 259.

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 must 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 by way of 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 and IRRVAF01, 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:

- “Using the exit for password quality control” on page 271
- “Allowing access when RACF is inactive” on page 283
- “Protecting the user's resources from the user” on page 283
- “Controlling access of shared user IDs” on page 284
- “Controlling access of shared user IDs” on page 296

Summary of installation-exit callers

Table 19 on page 250 and Table 20 on page 252 list the macros, commands, and utilities that give control to each exit routine.

Table 19. RACF installation-exits cross-reference table—Part 1 of 2

Functions	ICHNCV00	ICHRDX01 ICHRDX02	ICHRCX01 ICHRCX02	IRRACX01 IRRACX02	ICHRIX01 ICHRIX02	IRRVAF01 VALREXX
ADDGROUP						Note 4
ADDSD	X (Note 1, Note 2)	Note 1	Note 2			Note 4
ADDUSER						Note 4
ALTDSD			Note 2			Note 4

Table 19. RACF installation-exits cross-reference table—Part 1 of 2 (continued)

Functions	ICHNCV00	ICHRDX01 ICHRDX02	ICHRCX01 ICHRCX02	IRRACX01 IRRACX02	ICHRIX01 ICHRIX02	IRRVAF01 VALREXX
ALTGROUP						Note 4
ALTUSER						Note 4
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			Note 4
RDEFINE		Note 1	Note 2			Note 4
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	
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.

4. When a field in the CSDATA segment is specified, the exit gets control after each suboperand of the CSDATA keyword has been parsed according to the rules for the field specified in the CFDEF segment in the corresponding CFIELD profile.

Table 20. RACF installation-exits cross-reference table—Part 2 of 2

Function	ICHRLX01 ICHRLX02	ICHRFX01 ICHRFX02 ICHRFX03 ICHRFX04	ICHPWX01	ICHPWX11	ICHCNX00	ICHCCX00	IRREVX01
ADDGROUP							X
ADDSD					X		X
ADDUSER				X			X
ALTDSD					X		X
ALTGROUP							X
ALTUSER			X	X			X
DELDSD					X		X
DELGROUP						X	X
DELUSER						X	X
LISTDSD					X		X
PASSWORD			X	X			X
PERMIT					X		X
RALTER	Note 5	Note 6					X
RDEFINE	Note 5	Note 6			X		X
RDEFINE FROM (on data sets)							X
RDELETE							X
REMOVE						X	X
RLIST							X
SEARCH					X		X
SETROPTS RACLIST	Note 7						X
RACROUTE REQUEST= DEFINE							
RACROUTE REQUEST= AUTH							
RACROUTE REQUEST= VERIFY			X	X			
RACROUTE REQUEST= LIST	X						
RACROUTE REQUEST= FASTAUTH		X					
RACROUTE REQUEST= EXTRACT					X		
RACROUTE REQUEST= SIGNON							
IRRUT100					X		

Table 20. RACF installation-exits cross-reference table—Part 2 of 2 (continued)

Function	ICHRLX01 ICHRLX02	ICHRFX01 ICHRFX02 ICHRFX03 ICHRFX04	ICHPWX01	ICHPWX11	ICHCNX00	ICHCCX00	IRREVX01

Note:

5. 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.

6. 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.

7. 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 19 on page 250 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, ENVROUT, and NESTED=YES 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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) under the description of 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” on page 254). 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.

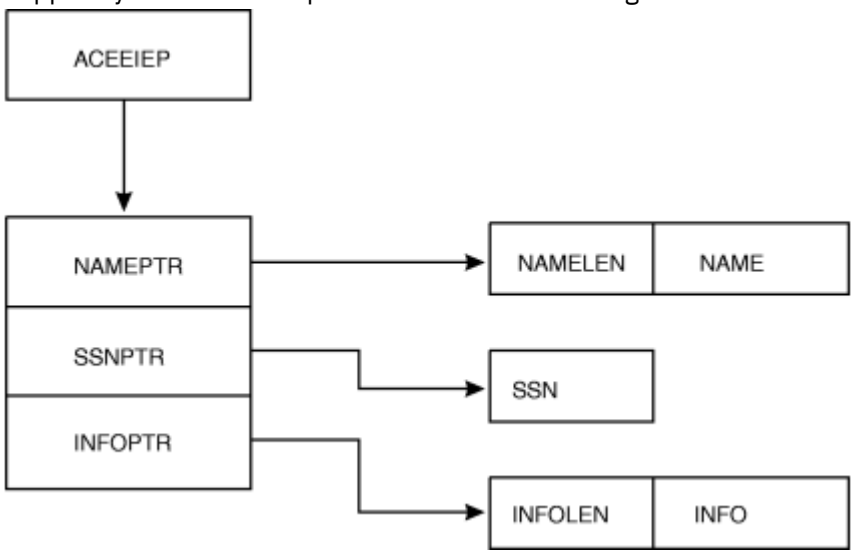
At expansion time, the process is reversed. The exit gets the storage for the original data structure, establishes appropriate pointers, and copies the data indicated by the range table into the storage.

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 when an ACEE is expanded from VLF or an ENVR object. 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 in SYS1.MODGEN. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 in SYS1.MODGEN. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 249.

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 250 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 21 on page 260 for a summary of these parameters.

Table 21. 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

Table 21. ICHCNX00-exit parameter processing (continued)

CALLER		OFFSET											
		0	**4**	**8**	**12**	**16**	**20**	**24**	**28**	**32**	**36**	**40**	**44**
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

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.

Note:

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.

Note:

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 250 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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
- RACMAP
- RACPRIV
- 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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

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
- Request that the command run with system SPECIAL or system AUDITOR authority.

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 by way of 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 about the dynamic exits facility, see *z/OS MVS Installation Exits*. For information about the PROG*xx* member of SYS1.PARMLIB, see *z/OS MVS Initialization and Tuning Reference*. For

information about the SETPROG EXIT command, see *z/OS MVS System Commands*. For information about 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 that is 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.

- Quoted text strings, such as values for the ADDUSER NAME keyword, appear as entered. Note that quoted text strings might be longer than allowed because of TSO parse processing. These strings are typically truncated in the RACF database.
- 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 uppercase.
- If the user's TSO profile is not set to NOPREFIX, the value of the PREFIX is inserted as the high-level qualifier for unquoted data sets. This value can be set to whatever the caller wants, using TSO *cmd* PROFILE PREFIX(value). The typical value is the user ID. However, often a group of users set a common prefix and that is used.
- 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 by prompting.
– An extra 300 bytes of blanks following the last keyword in the buffer, where the exit can add more 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.

    Note: Modifying fields in the ACEE may result in message IRR421I being issued by RACF when the ACEECHK class is active. The output parameter field can be used to request that the command run with system SPECIAL or system AUDITOR authority. The use of this mechanism will avoid message IRR421I.

    For information about the ACEE fields, see *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

  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.
- A pointer to an output parameter area, in which the exit can set bits to affect aspects of RACF command processing, such as running the command with increased authority.

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.
- Specify that the command should run with system SPECIAL or system AUDITOR authority (RACF removes the authority when the command completes. The postprocessing exit has no responsibility in this regard.)

  Note: This function is not supported for managed ACEEs. A managed ACEE is one created using the initACEE callable service (IRRSIA00) with the INTA_MANAGED option.
- Determine whether the command executes or fails, by setting a return code in register 15, as described in “Registers at exit” on page 268. 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 268. 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 re-parses 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 (although this is not recommended) 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 in SYS1.MODGEN. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 have SPECIAL authority until you deactivate the exit routine.

Note: Since IRREVX1B was first made available, RACF has been enhanced to provide a helpdesk function, as well as to provide a mechanism by which the exit can request that a command run with system SPECIAL authority. Thus, this sample is largely obsolete, but remains as an example of coding 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 250 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.

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.
- **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.

z/OS Security Server RACF Data Areas in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) contains a mapping of the exit parameter list, PWXP, which is mapped by macro ICHPWXP in SYS1.MODGEN.

Table 22 on page 270 shows which fields are available to the exit when the exit is called from the different RACF components.

Table 22. 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	—	—
28	Current password	X	—	O[1]
32	Password last change date	X	—	O[1]
36	ACEE	X[2]	X	X
40	Group name	O	—	—
44	Installation data	O	—	—
48	Password history	X	—	O[3]
52	Flag byte	X	—	—
56	Password last change date	X	—	X

X
means the field is always available.

O
means the field might be available.

—
means the field is never available.

Note:

1. Available only if NEWPASS is available.
2. Although available, the ACEE might not be fully initialized.
3. Available only if NEWPASS is available and KDFAES is not enabled.

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. For RACROUTE REQUEST=VERIFY, if register 15 contains any other values, processing ends with an abend. For ALTUSER and PASSWORD, if register 15 contains any other values, the request fails.

Using the exit for 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.

Coded example of the exit routine

IBM provides a sample new-password exit on the Internet. IBM does not support this sample. For information on how to get this sample from the RACF home page (www.ibm.com/us-en/marketplace/resource-access-control-facility-racf) or via anonymous FTP, see the RACF download page (github.com/IBM/IBM-Z-zOS/tree/master/zOS-RACF/Downloads).

New-password-phrase exit (ICHPWX11)

A password phrase is an alternative to a password that allows a longer length and a larger character set. RACF supports password phrases from 9 to 100 characters in length, made up of mixed case letters, numbers, and special characters, including blanks. When the new-password-phrase exit (ICHPWX11) is present and allows it, the password phrase can be 9–100 characters. When ICHPWX11 is not present, the password phrase must be 14–100 characters.

RACF enforces a basic set of rules for password phrases:

- Maximum length: 100 characters
- Minimum length:
  - 9 characters, when the encryption algorithm is KDFAES or ICHPWX11 is present and allows the new value
  - 14 characters, when ICHPWX11 is not present and the encryption algorithm is not KDFAES
- The user ID (as sequential uppercase characters or sequential lower case characters) is not part of the password phrase
- At least 2 alphabetic characters are specified (A - Z, a - z)
- At least 2 non-alphabetic characters are specified (numerics, punctuation, special characters, blanks)
- No more than 2 consecutive characters are identical

The installation has the option of using the new-password-phrase exit to augment RACF function when validating a new password phrase.

RACROUTE REQUEST=VERIFY processing and the ADDUSER, ALTUSER, PASSWORD, and PHRASE commands invoke the installation-supplied new-password-phrase processing exit. The exit gains control when a new password phrase is processed, and can examine the value specified for the password phrase and enforce installation rules in addition to the RACF rules. For example, while RACF does not allow the user ID to be part of the password phrase, the exit could perform more complex tests to also disallow the company name, the names of months, and the current year in the password phrase.

The use of the new-password-phrase exit augments the RACF rules, but cannot override them. Be sure that the exit and the RACF rules do not contradict each other. For example, if the exit requires that password phrases contain all alphabetic characters, users will not be able to create new password phrases because RACF requires at least two non-alphabetic characters.

The interval value specified on the PASSWORD command applies to both passwords and password phrases. It is processed by the new password exit, ICHPWX01, and is not passed to this exit

In a remote sharing environment, if password synchronization or automatic password direction is active, and a password phrase is changed, the new-password-phrase exit is always invoked on the node where the initial password phrase change is made. When RACF automatically updates the password phrase on other nodes, the new-password-phrase exit might or might not be invoked:

- If the password phrase was changed by a RACF command, and the command is propagated to another node by automatic command direction, the new-password-phrase exit is invoked on that node.
- If the password phrase was changed by other means (at logon, or by a RACROUTE or ICHEINTY invocation), and the password phrase change is propagated to another node by automatic password direction or password synchronization, the new-password-phrase exit is not invoked on that node.

Installing the exit routine

To install an installation-provided ICHPWX11 exit, name the exit ICHPWX11, load it into the link pack area (LPA), and re-IPL.

IBM provides a sample ICHPWX11 exit routine, and the REXX exec IRRPHREX that it invokes, in SYS1.SAMPLIB. The source for the sample ICHPWX11 exit routine is shipped in the RACEXITS member of SYS1.SAMPLIB. The corresponding load module for ICHPWX11 is shipped in SYS1.LINKLIB.

To use the sample exit routine that IBM provides, do the following:

1. Copy the REXX exec from member IRRPHREX in SYS1.SAMPLIB to the REXXLIB concatenation.
2. Install the exit 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 ICHPWX11 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.
   - Create an SMP/E USERMOD to move ICHPWX11 into LPALIB.
3. Re-IPL.

After you install the ICHPWX11 exit routine and IPL:

- If you change the password-phrase quality rules that are coded in the IRRPHREX exec, you need not re-IPL. The changes you make to IRRPHREX take effect immediately when you save them.
- If you make changes to ICHPWX11, you must re-IPL to activate your changes.

Exit routine environment

ICHPWX11 receives control in the following environment:

- When called from RACROUTE REQUEST=VERIFY processing:
  - In supervisor state
  - Under protection key 0
- When called from the ADDUSER, ALTUSER or PASSWORD commands:
  - If the command originates from a TSO user:
    - In problem state
    - Under protection key 8
    - In an APF-authorized environment
  - If the command is a directed command:
    - In supervisor state
    - Under protection key 0
  - If the command originates from the operator's console:
    - In problem state
    - Under protection key 2
    - In an APF-authorized environment
  - If the command was issued under another task, the state depends on the attributes of that task.
- It can have any RMODE, but should use AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

Exit routine processing

The ICHPWX11 exit is invoked in the following ways:

- **Through RACROUTE REQUEST=VERIFY processing**

  If a new password phrase is to be processed, REQUEST=VERIFY performs the following functions after invoking ICHRIX01 (if ICHRIX01 is present):

  - Validates the new password phrase for compliance with RACF's password phrase rules
  - Verifies that the new password phrase differs from the current password phrase
  - If the SETROPTS PASSWORD(HISTORY) option is active, verifies that the new password phrase differs from the previous password phrases

If the password phrase passes these checks, REQUEST=VERIFY invokes ICHPWX11.

- **Through the ADDUSER command**

  After parsing the command and checking the user's authorization, if the PHRASE keyword is specified ADDUSER validates the new password phrase for compliance with RACF's password phrase rules and invokes ICHPWX11.

- **Through the ALTUSER command**

  After parsing the command and checking the user's authorization, if the PHRASE keyword is specified ALTUSER validates the new password phrase for compliance with RACF's password phrase rules and invokes ICHPWX11.

- **Through the PASSWORD or PHRASE command**

  If the PHRASE keyword is specified, the command performs the following functions:

  – Validates the new password phrase for compliance with RACF's password phrase rules
  – Verifies that the new password phrase differs from the current password phrase
  – If the SETROPTS PASSWORD(HISTORY) option is active, verifies that the new password phrase differs from the previous password phrases

  If the password phrase passes these checks, the command invokes ICHPWX11.

Programming considerations

The new-password-phrase exit must be named ICHPWX11.

Code the ICHPWX11 exit to be reentrant. It can have any RMODE but should use AMODE(31) or AMODE(ANY) for the best use of virtual storage and best RACF performance.

The exit can run in the RACF subsystem address space. For more information, see “Exits running in the RACF subsystem address space” on page 250.

Entry specifications

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 list contents

Register 1 contains a pointer to the exit parameter list, PWX2, which is mapped by macro ICHPWX2 in SYS1.MODGEN. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for a mapping of the PWX2 data area.

Not all parameters in the parameter list are available to all invokers. Table 23 on page 275 summarizes which parameters are available depending on the RACF component that invoked the exit.

Table 23. Availability of parameters during ICHPWX11 processing for each RACF component that can invoke the exit

Offset (Decimal)	Parameter	REQUEST= VERIFY	ADDUSER	ALTUSER	PASSWORD
0	Length address	Always	Always	Always	Always
4	Caller address	Always	Always	Always	Always
8	Command processor parameter list address	Never	Always	Always	Always
12	New password phrase address	Always	Always	Always	Always
16	User ID address	Always	Always	Always	Always
20	Work area address	Always	Never	Never	Never
24	Current password phrase address	Sometimes. When the PASSWRD parameter specifies a PassTicket, the PHRASE parameter can be null. In this case, the parameter is not available to the invoker. In all other cases, it is available.	Never	Never	Always
28	password phrase last change date address	Always	Never	Never	Always
32	ACEE address	Always, but might not be fully initialized	Always	Always	Always
36	Group name address	Sometimes	Never	Never	Never
40	Installation data address	Sometimes	Never	Never	Never

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
One of the following return codes:

Value
Meaning

0
The new password phrase field is copied back into the calling function. Continue with processing.

4
The new-password-phrase request is not accepted and is to be failed. RACROUTE REQUEST=VERIFY processing terminates with a return code indicating a new password phrase that is not valid. The ADDUSER and ALTUSER commands ignore the request and continue processing. The PASSWORD command terminates processing.

8
The new-password-phrase 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 ADDUSER, ALTUSER, and PASSWORD commands are suppressed because the exit itself has already issued an appropriate message.

Note:

1. For RACROUTE REQUEST=VERIFY, return codes 0 and 4 are valid; if register 15 contains any other values, processing ends with an abend.
2. For the ADDUSER, ALTUSER, and PASSWORD commands, return codes 0, 4, and 8 are valid; if register 15 contains any other value, it is treated like return code 4.

Coded example of the exit routine

SYS1.SAMPLIB contains the source for a sample ICHPWX11 exit routine and the REXX exec IRRPHREX that ICHPWX11 invokes. (See "Installing the exit routine" on page 272.) The load module for ICHPWX11 is also included in SYS1.LINKLIB.

Password authentication exits

When KDFAES is not active

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, password phrase, 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. For more information see "The two-step method of password authentication" on page 58.

For more information on password authentication options see "Password authentication options" on page 56.

To use an installation-provided method of user verification, code the algorithm in the exit. For an encrypt operation, set the return code in the ICHDEX01 exit to 0. As a result, RACF stores the encrypted value that is returned by the exit. For a compare operation, set the return code to 0 if the comparison is successful, and set the return code to 12 if it is unsuccessful. 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. Provide an ICHDEX11 exit that sets the same return code.

If you do not provide an ICHDEX01 exit and activate it as described in "Installing the exit routine" on page 277, RACF uses the DES algorithm.

Note: In releases prior to V2R2, RACF uses the two-step method of checking described in The two-step method of password authentication. If you have an ICHDEX01 exit that sets the return code to 8 (to tell RACF to use only DES), the exit can be removed starting with V2R2.

When KDFAES is active

When KDFAES is active, the RACF manager no longer calls ICHDEX01 for encrypt operations against passwords, password phrases, or history values. RACF continues to call the exit for the compare operation for a user's password or password phrase until the user first changes it after KDFAES was enabled. The exit continues to be called for comparisons against history entries that are in the previous format. If the exit returns any return code other than 0 or 12 (that is, the exit made its own determination as to the validity of the field), or if the exit does not exist, RACF uses DES, but never masking, to determine whether the password is valid.

Once a password or password phrase is changed under KDFAES, the exit is no longer called for that user's password or phrase.

ICHDEX01

Installing the exit routine

There are two methods you can use to install your ICHDEX01 exit:

- Use an IEALPA*xx* member in SYS1.PARMLIB to request that MVS load ICHDEX01 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 ICHDEX01 in LPALIB.

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
- With the SYSZRAC2 ENQ held using SMC=STEP.

Exit routine processing

The ICHDEX01 exit is called by the RACF manager, RACROUTE REQUEST=VERIFY and RACROUTE REQUEST=VERIFYX 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. Additionally, a manager call is made from the RACF subsystem address space during RACF remote sharing password synchronization processing to update the password on the target node.

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, call the RACF manager or call other MVS services that might use ENQ.

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 ICHDEXP in SYS1.MODGEN. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 ICHDEXP in SYS1.MODGEN. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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.

Note:

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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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.
- If the user has access to the resource because of the WARNING option in the profile, the final RACF return code is 4.

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; start 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 that is 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 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 that are 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 grants or 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 map 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 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 helps to identify the user of a shared user ID in the cases where a security context (ACEE) was created from a certificate through certificate name filtering or hostid mapping. The X500 name is meaningful for auditing purposes only.

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 the description of the RDXP parameter list in *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary). 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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.

Hex	(Decimal)	Meaning
8	(8)	The request is accepted. No more processing is performed and the post-processing exit is bypassed.
C	(12)	The request is accepted. Processing continues, including the post-processing exit, but authorization checking is bypassed.

Note:

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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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.

Note:

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, FSACCESS or FSEXEC class, the FASTAUTH service is called directly from a callable service, and the SAF router exit, ICHRTX00, is not called.
3. The exits can view the values for the AUTHCHKS and CRITERIA keywords, but should not modify them.

Preprocessing exits (ICHRFX01 and ICHRFX03)

There are two RACROUTE REQUEST=FASTAUTH preprocessing exits. In general, ICHRFX01 is used for non-cross-memory calls and ICHRFX03 is used for cross-memory calls.

Exceptions: ICHRFX03, if present, is always called instead of ICHRFX01, even in non-cross-memory mode, in the following situations:

- A FASTAUTH request is invoked for the UNIXPRIV , FSACCESS or FSEXEC class.
- The ACEEALET or ENVRIN operand is specified.
- A supervisor state or system key caller provides a nested ACEE on a FASTAUTH request. It does not matter whether the nested ACEE is processed; for example, if the client is authorized or the resource is not delegated, ICHRFX03 is still called. For information about nested ACEEs and delegated resources, see Defining delegated resources in *z/OS Security Server RACF Security Administrator's Guide*.

The preprocessing exits are entered before the RACROUTE REQUEST=FASTAUTH service routine performs authorization checking.

Note: Although RACF might perform two authorization checks for a nested ACEE, ICHRFX03 is only called once, before either check occurs.

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 all registers that 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.

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 mapping or the FAST mapping in *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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'.

Upon return, 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 is not requesting FASTAUTH to audit this request
- 4 if the exit is requesting FASTAUTH to audit regardless of other auditing options set. (See the ASIS value of the LOG= parameter.)

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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary)). It points to the ICHRFX01 parameter list, which is mapped by RFXP or FAST.

When the ACEEALET 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'.

Upon return, 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 is not requesting FASTAUTH to audit this request.
- 4 if the exit is requesting FASTAUTH to audit regardless of other auditing options set. (See the ASIS value of the LOG= parameter.)

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. Figure 38 on page 291 shows the logic that RACF uses to determine which exit to call.

Note: For a nested ACEE, although two authorization checks might be internally driven, ICHRFX04 is only called once, after both checks have completed. It does not matter whether the nested ACEE is processed; for example, if the client is authorized or the resource is not delegated, ICHRFX04 is still called. For information about nested ACEEs, see Defining delegated resources in *z/OS Security Server RACF Security Administrator's Guide*.

```
if cross memory mode, or ACEEALET or ENVRIN or CRITERIA is specified,
   or the class is UNIXPRIV, FSACCESS, FSEXEC,
   or a nested ACEE is provided by a supervisor state or system key caller

  then

   only call ICHRFX04

else

   if RACLISTed by RACROUTE REQ=LIST, GLOBAL=YES or RACLISTed by SETR RACLIST
      or the class is in the dynamic class descriptor table

   then

      call ICHRFX04 first and then call ICHRFX02

   else

      only call ICHRFX02
```

Figure 38. Logic that determines whether ICHRFX02 or ICHRFX04 is called

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:

Conditions	**Processing sequence**
Regardless of how the class is RACLISTed: • Cross-memory, or • The ACEEALET keyword is specified, or • The ENVRIN keyword is specified, or • The CRITERIA keyword is specified, or • The UNIXPRIV class, or • The FSACCESS class, or • The FSEXEC class, or • A nested ACEE	1. ICHRFX03 2. Auth processing 3. ICHRFX04 4. Auditing
Non-cross-memory and the class is RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=NO	1. ICHRFX01 2. Auth processing 3. ICHRFX02 4. Auditing

Conditions	Processing sequence
Non-cross-memory, and the class is in the dynamic class descriptor table or the class is RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES or by SETROPTS RACLIST Note that 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" on page 292.	1. ICHRFX01 2. Auth processing 3. ICHRFX04 4. Auditing 5. 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 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 all registers that 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 2nd word contains a pointer to the class descriptor table entry used for authorization checking. (The exit must not change the contents of the class descriptor table entry.)

   – The 6th word contains a 2-byte profile type followed by a 2-byte profile length.

   The profile type contains one of the following values.

   0
   No information, because profile information was provided to the ICHRFX04 exit.

1
No profile used.

2
Discrete profile name in external format.

3
Generic profile name in internal format.

6
No profile was found.

The profile length is the length of the profile name (contained in the 7th word).

- The 7th word contains a pointer to the profile name used in the authorization check. The type and length of the profile name is contained in the 6th word.
- The 11th word indicates the authority used to determine authorization. The format of the first two bytes of the 11th word is:

  1... ....
  Reserved for IBM's use.

  .1.. ....
  Normal authority was used.

  ..1. ....
  OPERATIONS authority was used.

  ...1 ....
  Trusted authority was used.

  .... 1...
  Privileged authority was used.

  .... .000
  Reserved for IBM's use.

  0000 0000
  Reserved for IBM's use.
- 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, which is documented in *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).
- 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.

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 mapping or the FAST mapping in *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary). 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. Other information in the FXAP includes the address, type, and length of the profile name used in the authorization check, and the authority (for example, OPERATIONS or trusted) that was used to determine authorization.

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 2nd word contains a pointer to the class descriptor table entry used for authorization checking. (The exit must not change the contents of the class descriptor table entry.)
     - The 11th word contains an indicator in the high-order bit (bit 0):
       - If the bit is on, the access check was based on the authority of the nested user (the daemon) because the ACEE was nested, the resource was delegated, and the primary user (the client) did not have access.
       - If the bit is off, the access check was not based on the authority of a nested user because the ACEE was not nested, or the resource was not delegated, or the ACEE was nested and the primary user (the client) had access.

       For information about nested ACEEs and delegated resources, see Defining delegated resources in *z/OS Security Server RACF Security Administrator's Guide*.
     - 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 (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 ICHRFX04 parameter list is mapped by FXAP. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary). The ICHRFX04 parameter list points to the ICHRFX02 parameter list, which is mapped by RFXP or FAST.

When the ACEEALET 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 map 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 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 helps to identify the user of a shared user ID in the cases where a security context (ACEE) was created from a certificate through certificate name filtering or hostid mapping. The X500 name is meaningful for auditing purposes only.

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 might 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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 or password phrase 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 or password phrase, 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*.

When the user ID passed to RACROUTE REQUEST=VERIFY begins with ** (X'5C5C'), an identity context reference is being passed instead of a user ID and password. RACF calls the R_cacheserv SAF callable service to map the identity context reference to a user ID known to the RACF domain, and builds an ACEE using information from the identity context cache. The mapping of the identity context reference to a RACF user ID occurs before the preprocessing exit (ICHRIX01) is invoked. The user ID field in the exit parameter list (RIXUID) is set to the RACF user ID, the password field (RIXPWD) is set to zero, and the identity context extension field (RIXICTX) is set to point to an identity context extension (ICTX). The ICTX contains information about the original user that RACF includes in audit records; at the successful completion of the VERIFY request it will be anchored in the ACEE by the field ACEEICTX. Because an identity context reference identifies a user who has already been authenticated, the flag RIXPSCKN in the

exit parameter list is set to indicate that password checking should be bypassed, as if PASSCHK=NO was specified on the RACROUTE.

Note: When RACROUTE REQUEST=VERIFY is invoked with SYSTEM=YES, the exits will only get control if an ACEE was not expanded from VLF or an ENVR object.

When an identity context reference is passed, RACF does not use the following keywords in its subsequent processing, and the exit parameter list does not contain values for them even if they were specified on the RACROUTE request:

- JOBNAME
- SGROUP
- SUSERID
- SNODE
- EXENODE
- STOKEN
- REMOTE
- START

An ICTX can also be provided by the ICTX= keyword on the RACROUTE REQEST=VERIFY input parameter list. If it is provided both on the parameter list and resolved from an identity context reference (ICR), the one resolved from the ICR is the one used by RACF and passed to the exit in RIXICTX.

The exit must not free the ICTX area or change its length (ICTXLEN), but it can change the fields within the ICTX by changing the lengths and contents of the fields within the bounds of the existing area. It can delete fields by setting the field lengths to 0. If RIXICTX=0 on entry, it can provide a new ICTX block as described by the ICTX= keyword on the RACROUTE request, but it or the requestor is responsible for freeing the area in the event the request fails and an ACEE is not built that anchors the ICTX.

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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) contains a mapping of the VERIFY(X) request exit parameter list, RIXP, and a mapping of the identity context extension area, ICTX.

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 does not 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. You should 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 in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) contains a mapping of the RACROUTE REQUEST=VERIFY(X) exit parameter list, RIXP.

Note: If the IDTA keyword is specified and the IDTDATA class is active, the ICHRIX02 return code 4 is not valid. In this case, RACROUTE REQ=VERIFY will fail with the 8/70/6 return codes.

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 might 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 might 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 can be called multiple times. The exit is invoked in problem state key 8.

z/OS Security Server RACF Data Areas in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) 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.

Custom Field Validation Exits

There are two options for writing a validation exit:

1. The IRRVAF01 exit. This is an assembler exit that uses the dynamic exit facility. For more information, see “Custom Field Validation Exit (IRRVAF01)” on page 303.
2. A System REXX exec. The name of the exec to be used for a given custom field is specified in the custom field's definition. For more information, see “Custom Field Validation Exit (VALREXX)” on page 306.

These two mechanisms can coexist. IRRVAF01 is called first. If that succeeds, the designated REXX exit is called next.

Custom Field Validation Exit (IRRVAF01)

The IRRVAF01 exit provides an exit point for the ADDUSER and ALTUSER commands, ADDGROUP and ALTGROUP commands, ADDSD and ALTDSD commands, and RDEFINE and RALTER commands when a field in the CSDATA segment is specified. The exit gets control after each suboperand of the CSDATA keyword has been parsed according to rules for the fields from the CFDEF segment in the corresponding CFIELD profile for the field specified.

Controlling the exit routine through the dynamic exits facility

IBM has defined the IRRVAF01 exit point to the dynamic exit facility. Therefore, you can update the exit without re-IPLing. You can associate your installation exit routine with the IRRVAF01 exit point by way of 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 IRRVAF1A to the IRRVAF01 exit point, add the following to the PROG*xx* member:

```
EXIT ADD
     EXITNAME(IRRVAF01)
```

```
      MODNAME(IRRVAF1A)
      STATE(ACTIVE)
```

Alternatively, from the console issue the command:

```
SETPROG EXIT,ADD,EXITNAME=IRRVAF01,MODNAME=IRRVAF1A
```

For more information about the dynamic exits facility, see *z/OS MVS Installation Exits*. For information about the PROG*xx* member of SYS1.PARMLIB, see *z/OS MVS Initialization and Tuning Reference*. For information about the SETPROG EXIT command, see *z/OS MVS System Commands*. For information about 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.

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 IRRVAF01 exit point.

Exit routine processing

RACF gives control to IRRVAF01 after each suboperand of the CSDATA keyword has been parsed according to rules for the fields from the CFDEF segment in the corresponding CFIELD profile for the field specified. Values for fields defined as hex fields will be passed to the exit as an even number of characters. For example, if the value entered was ABCD, the exit will be passed X'ABCD', and a value entered as ABC will be passed as X'0ABC'. If ABC is entered for a field defined as a character field, it will be passed as X'C1C2C3'.

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 on the local system. RACF then sends the command to the remote systems. If an IRRVAF01 exit exists on a remote system, RACF invokes that exit on the remote system.

Information passed in the parameter list

The parameter list that is passed to the exit is mapped by macro ICHCDXP and contains fields such as:

- Address of data area
- Address of profile length and name. If multiple profile names are specified, the first one is passed to the exit, and the rest will be available in the command buffer.
- Address of class name

- Address of TSO/E CPPL containing address of command buffer, which contains:
  - An image of the original command after parsing.
  - Quoted text strings, such as values for the ADDUSER NAME keyword, appear as entered.

    Note:

    Quoted text strings might be longer than allowed because of TSO parse processing. These strings are typically truncated in the RACF database.
  - If the AT or ONLYAT keyword was specified, it does not appear in the command buffer.
  - The defaults for command keywords that have defaults and were not specified on the original command.
  - Any data that was provided by the prompting.
- Segment name
- Keyword name
- Flags
- A pointer to a 200 byte message area. If the exit fails the command with a message, it can provide message text in this area to be inserted into message IRR52217I. The exit should not change the pointer.

Note: Fields in the parameter list, and any values that it points to should not be changed, except for the 200 byte message area.

Based on the information that is passed to it in the parameter list, the exit routine can:

- Determine whether the keyword value is accepted or not, by setting a return code in register 15, as described in “Registers at exit” on page 306.
- Determine whether RACF issues message IRR52217I for a failed command, by setting a return code in register 15, as described in “Registers at exit” on page 306. The exit routine might provide text to be appended to message IRR52217I.

Programming considerations

Code the IRRVAF01 exit routine to be reentrant. Link-edit IRRVAF01 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, CDXP, which is mapped by macro ICHCDXP. See *z/OS Security Server RACF Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for a mapping of the CDXP data area.

Return specifications

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 IRRVAF01 exit point.

Registers at exit

Upon return from this exit, the register contents must be:

Register
Contents

0-14
Restored to contents at entry

15
One of the following return codes:

Value
Meaning

0
Continue processing the command.

4
The exit has failed the keyword value, and RACF is to re-prompt for the value if in prompt mode.

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.

Coded example of the exit routine

The RACEXITS member in SYS1.SAMPLIB contains a sample IRRVAF01 exit routine, IRRVAF1A. It shows the basic use of the fields in the parameter list and return code processing.

Custom Field Validation Exit (VALREXX)

As an alternative to using the IRRVAF01 exit, a REXX exec can be used to validate the value of a custom field as it is being set using the appropriate RACF command. The VALREXX operand in the CFDEF segment of a CFIELD class profile is used to optionally identify a System REXX exec to be called for validation. The exec must reside in the System REXX concatenation at the time the field value is assigned.

Note: VALREXX exits:

- are not identified in message IRR508I (which identifies active exits during system IPL)
- do not appear in the RACF exits report produced by the data security monitor (DSMON)
- do not require an IPL to add or change.

Exit routine processing

RACF gives control to the rexx exec after each suboperand of the CSDATA keyword has been parsed according to rules for the fields from the CFDEF segment in the corresponding CFIELD profile for the field specified.

The exec 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 on the local system. RACF then sends the command to the remote systems. If a rexx exec exit exists on a remote system, RACF invokes that exit on the remote system.

Input

The following arguments are passed to the exec, in the following order:

Table 24. REXX input values

Argument number	Purpose	Description
1	Command operation	ADD or ALTER
2	Class name	USER, GROUP, DATASET, or any general resource class
3	Profile name	Profile to which the field is being assigned. For DATASET, the name is fully qualified.
4	Segment name	CSDATA
5	Custom field name	The field name as defined in the CFIELD class
6	Custom field type	A single character: C - Character F – Flag (YES or NO) H - Hexadecimal N - Numeric
7	Custom field length	0 for numeric fields
8	Custom field value	Truncated to 512 characters. See the FullValue.*n* variable below. Values for fields defined as hexadecimal fields will be passed to the exit as an even number of characters. For example, if the value entered was ABC the exit will be passed 0ABC.
9	Generic profile indicator	0 or 1
10	Dataset volume	For class name DATASET only. For the ADDSD command, this is a blank-separated list of volume names specified on the VOLUME keyword. For the ALTDSD command, this is the single volume name specified on the VOLUME keyword to identify the discrete data set.

To account for argument values that may exceed 512 characters, the following variables are set by RACF when the validation exit is launched:

- FullValue.*n* – FullValue.0 contains the number of variables that follow. FullValue.1 – FullValue.*n* contain the custom field value split into 512-character-maximum pieces.
- OriginalCommand.*n* – OriginalCommand.0 contains the number of variables that follow. OriginalCommand.1 – OriginalCommand.*n* contain the command image split into 512-character-maximum pieces.

Notes:

1. If multiple profile names are specified, the first one is passed to the exit, and the rest are available in the command buffer.
2. If the AT or ONLYAT keyword is specified, it does not appear in the command buffer.

Output

The exec must exit with one of the following return values:

Return Value	Description
0	Success
4	Re-prompt the user if in prompt mode
8	Fail the command

Optional message text can be returned to RACF by setting the ExitMessage variable. This text will be appended to the end of IRR52223I when the exit returns with a non-zero value. The length cannot exceed 200 characters.

For example, if an exec named MYREXVAL returns with an exit value of 8, and sets ExitMessage to "MYFIELD must start with a digit and be followed by 9 characters.", the user would see the following message:

```
IRR52223I CSDATA validation failed by exit MYREXVAL. MYFIELD must start with a
digit and be followed by 9 characters.
```

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 8. Recovery procedures

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 about 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 about 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 313.

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 later 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 topic is further discussed in “Sysplex recovery scenarios that require XCF-local mode ” on page 318.

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.
Guideline: To avoid failsoft, use RVARY SWITCH rather than RVARY INACTIVE.

© Copyright IBM Corp. 1994, 2020

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.

Note: if an I/O error occurs on a data set of the primary RACF database, causing the device to be varied offline or boxed, and if the corresponding backup data set is active and on a device that has not been varied offline or boxed, RACF issues an RVARY SWITCH command to automatically switch to the backup data set without requiring the operator to enter a password.

Shared database considerations

The use of the RVARY command becomes more complex when the RACF database is shared with other systems. Generally, 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 need to issue the command only 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.

When an I/O error occurs on the RACF database and RACF does an automatic RVARY SWITCH to the backup database, the operator is not required to enter a password.

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 in data sharing mode)
- A SYSTEMS wide shared ENQueue (when RACF is in data sharing mode)

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 RESERVEs, 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 canceling 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, 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, and you need to deactivate the primary database, you must use the RVARY INACTIVE command. If your database has a single data set, this command 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 105 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 311, 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 12 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. Take one of the following actions:
   a. Issue RVARY SWITCH (the backup is now the *new primary*).

b. Vary offline the device that the primary resides on. RACF automatically does an RVARY SWITCH.

3. Take one of the following actions:
   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 option "3.b" on page 314 with IRRUT400 with LOCKINPUT, 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 option "3.b" on page 314, and use IRRUT400 with LOCKINPUT. But note that even in this scenario your databases could become out of synch between steps "6" on page 314 and "7" on page 314.

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. Take one of the following actions:
   a. Correct the problem on the backup, using BLKUPD.
   b. If the device is accessible, copy the primary onto the backup, using IRRUT200 with PARM=ACTIVATE 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. If you used IRRUT400, or IRRUT200 without PARM=ACTIVATE in option "2.b" on page 314, issue RVARY ACTIVE for the backup. If you used IRRUT200 with PARM=ACTIVATE, the backup is already active.
4. If you used option "2.b" on page 314 with IRRUT400 with LOCKINPUT, 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 option "2.b" on page 314 with IRRUT400 and LOCKINPUT or IRRUT200 with PARM=ACTIVATE.

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 112.
2. Obtain the most recent dump of your RACF database.
3. Take one of the following actions:
   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.

When 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.

If both the primary database and the backup database are updated and users are prevented from logging on, (for example, users are specifically revoked, the SETROPTS INACTIVE is set to one day, and a weekend is passed and all users revoked.), and if you prefer to fix your primary, instead of using the most recent offline backup of your database (which might need a "replay" of all valid commands since that backup was made), you might do the following.

1. Use an emergency data set name table, which names as the emergency primary a new RACF database, and, which names the production primary database as the emergency backup database. Make sure that you set the flag to duplicate updates for the backup.
2. Re-IPL.
3. Log on to IBMUSER. For example, all of your users in your production database are revoked, however, on the emergency primary, IBMUSER is not revoked, and is special.

   Note:

   If IBMUSER on your primary production database is revoked, you might unrevoke IBMUSER right away, or after step “4” on page 315.
4. Issue RVARY SWITCH and then the emergency primary is the production primary (and IBMUSER's ACEE is still special). Continue with whatever is appropriate for your situation (for example, unrevoke your users, change SETROPTS INACTIVE back to a reasonable value, and so on).
5. Resync your offline production backup database from your updated primary database with IRRUT200 or IRRUT400.

   Note:

   Since the production backup database is offline, IRRUT200's PARM=ACTIVATE is not applicable.
6. IPL with your normal data set name table to activate the real production primary and backup databases.

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.

Note: Do not issue a SETXCF command to force the rebuild of a structure into a coupling facility that is not available to the system. Do not issue the RVARY DATASHARE command when there is no coupling facility available. These actions cause the system to enter read-only mode. To exit read-only mode after an inappropriate SETXCF command, issue the RVARY DATASHARE command and RACF will use the original coupling facility. To exit read-only mode after an inappropriate RVARY DATASHARE command, or in other situations when no coupling facility is available, issue the RVARY NODATASHARE command.

Read-only mode

A system experiencing a problem using one or more RACF cache structures might enter read-only mode, with RACF issuing message IRRX004A. 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 IRRX004A, 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 the data set name table 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 318 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 data set name table 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 318 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 316 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 use the coupling facility by way of the IXLCACHE macro; IRRX016I is issued. One of two recovery actions occur automatically:

- 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” on page 318 for more information.
- 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 by way of 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" on page 318 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
IRRX004A
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.

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 112.) 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.

Guideline: Sysplex customers should use the TSO/E user attributes data set (UADS) to authorize at least one ID. This gives you a way to log on to TSO/E. But the system will have limited functionality due to the absence of RACF.

If a user logs on to TSO/E and you have not defined a TSO segment for that user, TSO/E checks the SYS1.UADS data set for the information it needs to build a session. If TSO/E 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 one system programmer is recommended). This allows you to log on to TSO/E when RACF is not up (failsoft).

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/E 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/E user ID should you need to bring a member up in XCF-local mode.

For further information on UADS, see:

- *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 data set name table with both the sysplex communication and data sharing bits off. "Emergency data set name tables" on page 49 describes how to set up such an emergency data set name table.

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.

Note:

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 315.
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
- RACMAP
- 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, processing 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 324. 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 z/OS 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” on page 324.

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 OS/390 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 323 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 OS/390 release 10 or higher. After all systems sharing the database are migrated to OS/390 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 323 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 by way of 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 IRRX004A. 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 6, “RACF database utilities,” on page 203.

      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 must 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 is 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 do not 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 is 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 83 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) 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 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” on page 332, 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 332. 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.)

Note:

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 85 and “Restarting the RACF subsystem” on page 82 for more information.

Chapter 9. Storage estimates

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 that is needed for a RACF database depends mainly on the number of users, groups, user-group connections, and resource profiles that are defined to RACF. Other factors that might affect the amount of space that is required for the database include:

- The length of the names of the entities that are defined to RACF
- Activating the RACGLIST class
- Activating the CACHECLS class
- How efficiently the space in the RACF database is used
- The password encryption algorithm in effect

Formula for the RACF database size

The formula to estimate the number of 4K (4096-byte) blocks required for a RACF database is

$$12 + A + B + B_A + C$$

where:

12
Is the number of blocks that are required for the ICB and database templates

A
Is the number of blocks that are required for the profiles (see “Calculating the number of blocks required for the profiles” on page 335)

B
Is the number of index blocks that are required (see “Calculating the number of index blocks required” on page 338)

B_A
Is the number of blocks that are required for the alias index (zero if you are running application identity mapping in stage 0) (see “Calculating the number of blocks required for the alias index” on page 338)

C
Is the number of BAM blocks that are required (see “Calculating the number of BAM blocks” on page 339)

Calculating the number of blocks required for the profiles

Use the formula that is shown in Figure 39 on page 336 to calculate the number of blocks that are required for the profiles (the value of A is described in “Formula for the RACF database size” on page 335).

© Copyright IBM Corp. 1994, 2020

```
 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  16   3  16  16  16   2  16  16   8  16  16  16  16  16

     + AA + BB + CC + DD + EE + FF + GG + HH + II + JJ + KK + LL + MM + NN + OO + PP + QQ + RR
+ SS
      ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---
---
       16    8   16   16   16   16   16    3   16   16    3   16   16    2   16   16   16
16   16
     + TT + UU + VV
      ---  ---  ---
       16   16   16
```

Figure 39. The number of blocks that are required for the profiles

In the formula,

F
= the number of users that are defined to RACF

G
= the number of groups that are defined to RACF

H
= the number of data sets that are defined to RACF

I
= the number of general resources that are defined to RACF

J
= the number of user CICS segments

K
= the number of user CSDATA segments

L
= the number of user DCE segments

M
= the number of user DFP segments

N
= the number of user EIM segments

O
= the number of user KERB segments

P
= the number of user LANGUAGE segments

Q
= the number of user LNOTES segments

R
= the number of user PROXY segments

S
= the number of user NDS segments

T
= the number of user NETVIEW segments

U
= the number of user OMVS segments

V
= the number of user OPERPARM segments

W
= the number of user OVM segments

X
= the number of user TSO segments

Y
= the number of user WORKATTR segments

Z
= the number of group CSDATA segments

AA
= the number of group DFP segments

BB
= the number of group OMVS segments

CC
= the number of group OVM segments

DD
= the number of group TME segments

EE
= the number of data set DFP segments

FF
= the number of data set TME segments

GG
= the number of general resource ALIAS segments

HH
= the number of general resource CERTDATA segments

II
= the number of general resource CFDEF segments

JJ
= the number of general resource DLFDATA segments

KK
= the number of general resource EIM segments

LL
= the number of general resource ICTX segments

MM
= the number of general resource KERB segments

NN
= the number of general resource PROXY segments

OO
= the number of general resource SESSION segments

PP
= the number of general resource STDATA segments

QQ
= the number of general resource SSIGNON segments

RR
= the number of general resource SVFMR segments

SS
= the number of general resource TME segments

TT
= the number of general resource CDTINFO segments

UU
= the number of general resource SIGVER segments

VV
= the number of general resource ICSF segments

Note:

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 factors such as installation data, the number of users in a group, and the number of entries on a data set or resource access list. 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. Set the values for Q, S, U, and BB 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.

Calculating the number of index blocks required

Use the formula that is shown in Figure 40 on page 338 to calculate the number of index blocks required. Use the result as the value of B in "Formula for the RACF database size" on page 335.

```
B = the number of index blocks required:
B = the 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 + PP + QQ + RR + SS + TT + UU + VV

E = the number of names that fit into an index block, which is approximately:

             4096 x .5
     -------------------------
     10 + average name length
```

Figure 40. The number of index blocks required

Note:

1. The values of F through Z and AA through VV are described in "Calculating the number of blocks required for the profiles" on page 335.
2. The formula for B is equivalent to:

   $$B = D/E^1 + D/E^2 + \ldots + D/E^9 + D/E^{10}$$

3. 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 you specify the average index name length.

Calculating the number of blocks required for the alias index

Use the formula that is shown in Figure 41 on page 339 to calculate the number of index blocks required. Use the result as the value of B_A in "Formula for the RACF database size" on page 335.

```
B_A = the number of blocks required for the alias index
    (zero if running application identity mapping in stage=0)

B_A ~ 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

       D_A
B_A ~  -----
       E_A

where:
        D_A = Q + S + U + BB + GG

                                                 4096 x 0.5
        E_A =  ------------------------------------------------------------
                  18 + average user or group name length + 3 + average alias name length

        18 = the length of the alias index entry overhead

         3 = the length of the alias index entry key prefix
```

Figure 41. The number of blocks that are required for the alias index

Note: The calculation for E_A makes the following assumptions:

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 you specify 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 for multiple groups to share a single GID, the implementation is not suggested.
3. The average alias name length is the average length of your SNAME, UNAME, GID, UID, and IPLOOK values.

Calculating the number of BAM blocks

Use the formula that is shown in Figure 42 on page 339 to calculate the number of BAM blocks required. Use the result as the value of C in “Formula for the RACF database size” on page 335. One BAM block is required for every 2038 blocks in the RACF data set.

Note: Round C up to a whole number.

```
C = the number of BAM blocks required.
C = 13 + A + B
    ----------
       2038
```

Figure 42. The number of BAM blocks required

Note: The values A and B are described in “Formula for the RACF database size” on page 335.

RACF virtual storage requirements

This table summarizes which functions give control to specified RACF exit routines.

Figure 43 on page 340 is a storage map for RACF. Table 25 on page 341 gives virtual storage requirements for RACF.

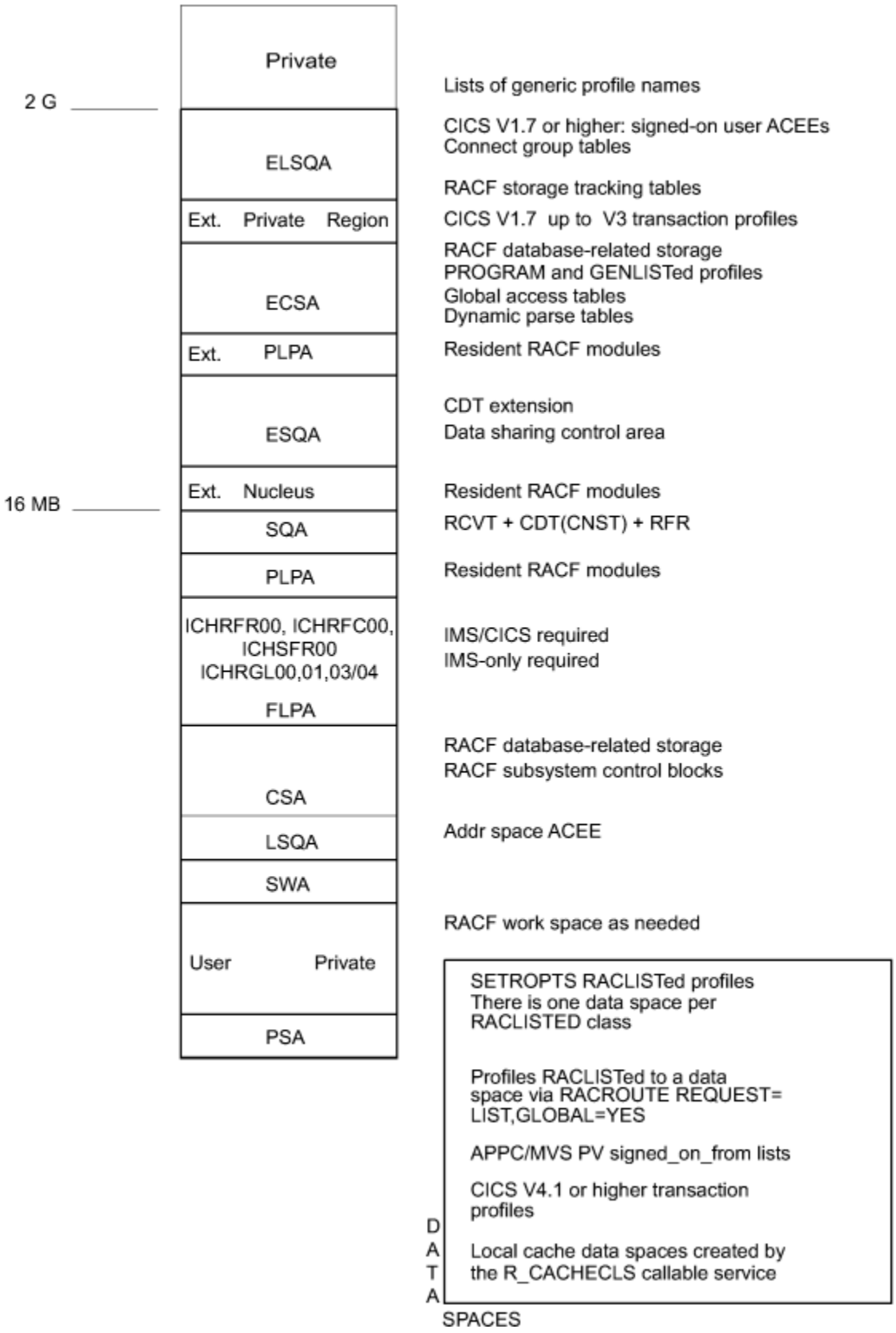


Figure 43. RACF storage use

Table 25. 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 by using AMBLIST
PLPA	RACF installation exits that are AMODE(24) or AMODE(ANY)	Measure by 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 by using AMBLIST
	RACF range table	4 + (*number_of_ranges* × 45)
EPLPA	RACF installation exits that are AMODE(31)	Measure by using AMBLIST
	RACF resident modules above 16MB	875,000
SQA	RACF communications vector table and extension	2800
	Class descriptor table (CNST)	7500 + 58 × *number_of_static_installation-defined_classes*
ESQA	RACF data sharing control area	300 (when enabled for sysplex communication)
	RACF token table	2408 bytes (when enabled for sysplex communication)
	Class descriptor table (CNSX)	(*number_of_classes_IBM_supplies* × 28) + (*number_of_static_installation-defined_classes* × 58) + 26 For z/OS V2R3, IBM supplies 238 classes, so the size of the CNSX is 6550 + (*number_of_static_installation-defined_classes* × 58). If you install a PTF that adds classes, you must recalculate this number.
	RACF identity cache communication vector (RCVI)	6880
LSQA	ACEE and related storage **Note:** 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 was dubbed a z/OS UNIX process, 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. If the user is identified by an identity context reference, add: 40 + *length_of_authenticated_user_name* + *length_of_registry_name* + *length_of_host_name* + *length_of_authentication_mechanism_OID*. The maximum value of the sum is 949.
ELSQA	Connect group table	64 + (48 × *number_of_groups_connected*)
	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 25. 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	27,640 + 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 you are using sysplex communication, for each backup data set add: 3248 + (4136 × *number_of_database_buffers* × 2)
	Dynamic parse tables	92,835 + (*number_of_custom_field_definitions* x 500)
	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
	RACF program verification module (IRRPVERS)	500,000
User private below 16MB	RACF transient storage	122 bytes while a RACF service is running
User private above 2G	Generic profile memory objects	Minimum of 2 MB for each generic profile list in use for the address space **Note:** 1. Large profile lists (containing thousands of profiles) might require 3 MB or more each. 2. This storage is not subject to the MEMLIMIT of the job and does not reduce the amount of storage available for the user's use.

Coupling facility cache structure storage requirements

See “Defining RACF structures for the coupling facility” on page 102 for information about how to calculate the storage that is required for coupling facility cache structures.

Appendix A. 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.

RRSF node configuration worksheet

Table 26. RRSF node configuration worksheet

Worksheet section	Worksheet labels and fields
Local Node:	Name: _____
Type of Node:	____ Single-system ____ Multisystem
Local node status:	____ Operative ____ Dormant
APPC protocol information:	LUNAME: _____ (from SYS1.PARMLIB(APPCPM*xx*)) TPNAME: _____ (default=IRRRACF) MODENAME: _____ (default=IRRMODE)
TCP/IP protocol information	ADDRESS: _____ (host name or IP address) PORTNUM: _____ (default=18136)
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)

© Copyright IBM Corp. 1994, 2020

Table 26. RRSF node configuration worksheet (continued)	
Worksheet section	**Worksheet labels and fields**
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)

Table 27. RRSF node status worksheet	
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.
  - The host name is MVS01.EXAMPLE.COM
  - You already have the following JCL to activate the RACF subsystem:

```
//RACF PROC
//RRSF EXEC PGM=IRRSSM00
//SYSOUT DD DSN=RRSF.OUTPUT,DISP=SHR
```

  - 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.
  - The host name is MVS02.EXAMPLE.COM
  - You already have the following JCL to activate the RACF subsystem:

```
//RACF PROC
//RRSF EXEC PGM=IRRSSM00
//SYSOUT DD DSN=RRSF.OUTPUT,DISP=SHR
```

  - 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:	`_X__ 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)
TCP/IP protocol information	ADDRESS: _MVS01.EXAMPLE.COM_____ (host name or IP address) PORTNUM: _____ (default=18136)
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:

```
_X__ Single-system   ____ Multisystem

                            System Name: _____________  Main: ___ Yes ___ No
```

Local node status:

```
_X__ Operative     ____ Dormant
```

APPC protocol information:

```
LUNAME:   _MF2AP002_____________________ (from SYS1.PARMLIB(APPCPMxx))
TPNAME:   _____________________________ (default=IRRRACF)
MODENAME: _____________________________ (default=IRRMODE)
```

TCP/IP protocol information

```
ADDRESS:  _MVS02.EXAMPLE.COM___________ (host name or IP address)
PORTNUM:  _____________________________ (default=18136)
```

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 (by way of 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 345 and "Completed RRSF node configuration worksheet for node MVS02" on page 347 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 using APPC:

```
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 is created for MVS01 on volume DASD01:

   - SYS1.RACF.SYSMVS01.INMSG
   - SYS1.RACF.SYSMVS01.OUTMSG
   - SYS1.RACF.MF1AP001.MF2AP002.INMSG
   - SYS1.RACF.MF1AP001.MF2AP002.OUTMSG

   If you wanted to communication link to use TCP/IP instead of APPC, the TARGET commands would be:

```
TARGET NODE(MVS01) DESCRIPTION('MEMPHIS MVS 1') -
  PREFIX(SYS1.RACF) LOCAL -
  WORKSPACE(VOLUME(DASD01)) -
  PROTOCOL(TCP(ADDRESS(MVS01.EXAMPLE.COM))) OPERATIVE
TARGET NODE(MVS02) DESCRIPTION('ORLANDO MVS PROD') -
  PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(DASD01)) -
  PROTOCOL(TCP(ADDRESS(MVS02.EXAMPLE.COM))) OPERATIVE
```

   and the following VSAM workspace data sets would be created for MVS01 on volume DASD01:

   - SYS1.RACF.SYSMVS01.INMSG
   - SYS1.RACF.SYSMVS01.OUTMSG
   - SYS1.RACF.SYSMVS01.MVS02.INMSG
   - SYS1.RACF.SYSMVS01.MVS02.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 using APPC:

```
TARGET NODE(MVS02) DESCRIPTION('ORLANDO MVS PROD') -
  PREFIX(SYS1.RACF) LOCAL -
  WORKSPACE(VOLUME(DASD02)) -
  PROTOCOL((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 is created for MVS02 on volume DASD02:

- SYS1.RACF.SYSMVS02.INMSG
- SYS1.RACF.SYSMVS02.OUTMSG
- SYS1.RACF.MF2AP002.MF1AP001.INMSG
- SYS1.RACF.MF2AP002.MF1AP001.OUTMSG

If you wanted to communication link to use TCP/IP instead of APPC, the TARGET commands would be:

```
TARGET NODE(MVS02) DESCRIPTION('ORLANDO MVS PROD') -
  PREFIX(SYS1.RACF) LOCAL -
  WORKSPACE(VOLUME(DASD02)) -
  PROTOCOL((TCP(ADDRESS(MVS02.EXAMPLE.COM))) OPERATIVE
TARGET NODE(MVS01) DESCRIPTION('MEMPHIS MVS 1') -
  PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(DASD02)) -
  PROTOCOL(TCP(ADDRESS(MVS01.EXAMPLE.COM))) OPERATIVE
```

As a result of running these TARGET commands, the following VSAM workspace data sets would be created for MVS02 on volume DASD02:

- SYS1.RACF.SYSMVS02.INMSG
- SYS1.RACF.SYSMVS02.OUTMSG
- SYS1.RACF.SYSMVS02.MVS01.INMSG
- SYS1.RACF.SYSMVS02.MVS01.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
   //SYSOUT DD DSN=RRSF.OUTPUT,DISP=SHR
   ```

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
   //SYSOUT DD DSN=RRSF.OUTPUT,DISP=SHR
   ```

6. Issue the RACF STOP command to shut down the RACF subsystem on node MVS01, using the locally defined RACF subsystem prefix.

   ```
   @STOP
   ```

7. Issue the RACF STOP command to shut down 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 is read by MVS and the module gets 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 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 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 are 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 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>
                 LISTENER STATUS  - ACTIVE
   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 127 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 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>
                 LISTENER STATUS  - ACTIVE
   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 127 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 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>
                 LISTENER STATUS  - ACTIVE
   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 127 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 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>
                 LISTENER STATUS  - ACTIVE
   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 127 for more details.

16. Enter the following SET command from MVS01 to list the attributes of the MVS01 node:

```
@SET LIST
```

You receive the following output:

```
IRRH005I (@) RAC1 SUBSYSTEM INFORMATION:
   TRACE OPTIONS                  - NOIMAGE
                                  - NOAPPC
                                  - NOSYSTEMSSL
                                  - NORRSF
                                  - NORACROUTE
                                  - NOCALLABLE
                                  - NOPDCALLABLE
                                  - NODATABASE
                                  - NOGENERICANCHOR
                                  - NOASID
                                  - NOJOBNAME
                                  - NOCLASS
                                  - NOUSERID
   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         - HRF77A0  00000194.00000022
         DYNAMIC PARSE VERSION    - HRF77A0
```

17. Enter the following SET command from MVS02 to list the attributes of the MVS02 node:

```
@SET LIST
```

You receive the following output:

```
IRRH005I (@) RAC2 SUBSYSTEM INFORMATION:
   TRACE OPTIONS                  - NOIMAGE
                                  - NOAPPC
                     - NOSYSTEMSSL
                                  - NORRSF
                                  - NORACROUTE
                                  - NOCALLABLE
```

```
                                  - NOPDCALLABLE
                                  - NODATABASE
                                  - NOGENERICANCHOR
                                  - NOASID
                                  - NOJOBNAME
                                  - NOCLASS
                                  - NOUSERID
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            - HRF77A0  00000194.00000022
      DYNAMIC PARSE VERSION       - HRF77A0
```

18. This two-node network is now configured for remote communication using APPC. 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 (by way of 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 B. 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 do not have a data set name table). It enables you to bypass RACF initialization processing (and RACF is inactive) and you can disallow duplicate names for discrete data set profiles. These options are typically 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).

  Guideline: Use the data set name table instead of the ICHSECOP module to control the number of resident data blocks. If you specify 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 so you cn 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	Bit	Description
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 for IBM's use
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.

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/E 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.

© Copyright IBM Corp. 1994, 2020

- 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. 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 a data set name table 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, PHRASE, NEWPHRASE, ICTX, ICRX, IDID, and IDTA keywords. The programs must be reentrant and fetched from an APF-authorized library.

Guideline: Because incorrect use of ICHAUTAB can cause system integrity problems, do not use ICHAUTAB. Instead run the 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.

If you have placed the names of any other programs into the authorized-caller table, you should protect them using the outlined approach. 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, PHRASE, NEWPHRASE, ICTX, ICRX, IDID, and IDTA keywords.

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.

Appendix C. MAIN switch procedure in a mixed-release network

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.

Restriction: You cannot change the main system to or from a node for which a non-functional TCP/IP protocol instance exists, or for which a protocol conversion is in progress. If you try to do this, the TARGET command fails.

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, use one of the following procedures depending on the release levels of the old and new main systems. A system is "downlevel" if it is running a release lower than z/OS V2R2, and "uplevel" otherwise.

The following scenarios should be considered.

Old or new MAIN is downlevel

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.

   CAUTION: It may be preferable to skip step 1 and start with step 2 at a quiet time with prior warnings to administrators.

2. On the original local main system, issue the **STOP** command to stop the RACF subsystem.

   The system will attempt to complete all work that is in progress and 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 system that is on a lower release than z/OS V2R2, issue **TARGET DORMANT** commands for the original and new main systems. Do not perform step 7 until the INMSG files for the original and new main systems on each remote node have drained.

     Tip: 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, IRROPTxx, containing the commands. Then you can issue a **SET INCLUDE(xx)** command on each system instead of manually entering the commands.

© Copyright IBM Corp. 1994, 2020

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 using the TARGET command" on page 159 for more information.
6. Issue the same **TARGET MAIN** command that you issued in step 5 on each nonmain system on the local multisystem node.
7. On each remote system that is on a lower release than z/OS V2R2, 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. On systems lower than z/OS V2R2, or on systems without SET FULLRRSFCOMM in effect, this command causes RACF to allocate INMSG and OUTMSG files for the new MAIN system, if they do not already exist. On systems lower than z/OS V2R2, this command causes RACF to transfer requests and returned output from the original main system's OUTMSG workspace data set to the new main system's OUTMSG workspace data set. On z/OS V2R2 systems, the work is not copied, but it will be sent in the proper order nonetheless when a connection is established with the new MAIN system.
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 on a lower release than z/OS V2R2 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 and, if necessary, TSO/E and JES on the original main system.

Old and new MAIN are uplevel

Follow these steps:

1. Make connections dormant:

- On each remote system that is on a lower release than z/OS V2R2, issue **TARGET DORMANT** commands for the original and new main systems. Do not perform step 4 until the INMSG files for the original and new main systems on each remote node have drained.

Tip: 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, IRROPTxx, 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.

2. Perform the switch between the old and new MAIN systems using the PLEXNEWMAIN (or NEWMAIN if not in a Sysplex) keyword of the **TARGET** command. Here, you are simply following the process described in "Reconfiguring a multisystem node" on page 174, which will differ depending on whether the multisystem node is in a sysplex or not.

   For example, if the multisystem node name is NODEABC, and MVSB is the new main system, and the multisystem node is in a sysplex, issue:

```
TARGET NODE(NODEABC) SYSNAME(MVSB) PLEXNEWMAIN
```

3. Issue the same **TARGET** command that you issued in step 2 on each downlevel nonmain system on the local multisystem node, substituting the MAIN keyword for the PLEXNEWMAIN (or NEWMAIN if not in a Sysplex) keyword.
4. Issue **TARGET LIST** commands to verify that the INMSG data sets on the remote downlevel 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** command that you issued in step 2, substituting the MAIN keyword for the PLEXNEWMAIN (or NEWMAIN if not in a Sysplex) keyword. This command causes RACF to allocate INMSG and OUTMSG files for the new MAIN system, if they do not already exist. Also, RACF transfers requests and returned output from the original main system's OUTMSG workspace data set to the new main system's OUTMSG workspace data set.
5. On each remote system that is on a lower release than z/OS V2R2 issue **TARGET OPERATIVE** commands for the original main (if it is to remain in the multisystem node) and new main systems.
6. 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** commands in steps 2, 3, and 4. This approach helps to ensure that no parameter library updates are missed.

Appendix D. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number `3.1`) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

© Copyright IBM Corp. 1994, 2020

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element

The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element

The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.

2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST,` and `2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

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

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

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 and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

© Copyright IBM Corp. 1994, 2020

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 document 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.

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

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)

- For information about currently-supported IBM hardware, contact your IBM representative.

RSA Secure code

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

Index

Special Characters

??? state 134
* (asterisk)
    default for started procedures 109
    in the data set name table 37
    in the started procedures module 110
&RACLNDE 194
&RACUID.**/ALTER entry in the global access table 32
=
    in the started procedures module 110

Numerics

382 abend code 282, 283
383 abend code 301, 302
385 completion code 286

A

abend codes
    382 282, 283
    383 301, 302
ACBNAME 161
access authority
    checking on job restarts 124
    checking with RACROUTE REQUEST=AUTH 281
    how it can cause accidental destruction of data 284
    started procedure 106
access method service commands
    IMPORT command 121
    IMPORTRA command 121
    LISTCAT command 121
    REPRO command 121
    RESETCAT command 121
    using with RACF-protected VSAM data sets 121
accessibility
    contact IBM 363
    features 363
accidental deletion of data
    preventing 283
ACEE
    when there are multiple users per address space 123
ACEE (accessor environment element)
    compression/expansion exit routines 253
    default built by RACROUTE REQUEST=VERIFY 299
    when ICHRIX01 is responsible for creating 301
ACEEIEP 253
ACEERASP bit 250
activating
    automatic direction 156
    RACF databases 311
    the RACF subsystem 77
active state for RRSF listener process 132
ADDRESS keyword on TARGET command 162
address space
    for sysplex data sharing 101
address space *(continued)*
    multiple users 123
    RACF subsystem
        stopping 85
    RACFDS 101
ADDVOL operand
    RALTER command
        moving multivolume tape data sets 123
administering security 2
administration, RACF
    classroom courses xv
ADRDSSU (DFDSS) system utility
    accessing a RACF-protected DASD data set 116
ADSP (automatic data set protection) attribute
    when using IEHMOVE 118
algorithm
    Data Encryption Standard (DES) 56
    masking 56
alias index blocks
    sample formatted output from IRRUT200 229
ALIGN keyword
    IRRUT400 utility 240
ALLOCATE system operation
    failures during 328
Allocating workspace files
    remote non-MAIN connections, for 157
allocation
    BAM/allocation comparison 229
allocation authorization checks
    using the RACROUTE REQUEST=DEFINE exit routine to accept 284
ALLOWINBOUND keyword on TARGET command 160
ALTER command, access method services
    renaming RACF-indicated data sets 117
ALTER system operation
    failures during 329
AMASPZAP service aid
    changing the ICHRSMFI default values 65
APPC considerations for an RRSF network 149
application identity mapping
    converting database for 204
    recovering from errors 324
    with shared RACF database 8
application updates, automatic direction 128
assistive technologies 363
association between user IDs
    overview 127
asterisk (*)
    in the data set name table 37
ASXBSENV pointer to ACEE 250
AT keyword on commands 128
AT-TLS
    sample policy 153
    setting up for RRSF 153
AT-TLS policy, sample for RRSF 135
attributes of an RRSF node, listing 155
AUDIT operand

AUDIT operand *(continued)*
ADDSD command 18
ALTDSD command 18
RALTER command 18
RDEFINE command
effect on system performance 18
SETROPTS command
effect on system performance 19
authorization checking
performing additional checks for users 299
using global access checking 32
authorized caller table 112
authorized-caller table (ICHAUTAB) 356
automatic command direction
activating and deactivating 156
commands issued from the RACF parameter library 185
automatic direction
activating and deactivating 156
automatic direction of application updates
activating and deactivating 156
automatic password direction
activating and deactivating 156
automatic step restart 124

B

backing up a RACF database 7
backout routines
commands that have 320
backup RACF database
creating 16
defining in the data set name table 17
description 6
performance impact 17
backup RACF databases
effect when inactive in a sysplex 102
levels of backup 17
BAM blocks
BAM/allocation comparisons 230
encoded map
codes used in 230
sample printout of an encoded map 232
basic RACF concepts 2
BLKUPD
to correct problems with the RACF database 327
BLKUPD command 245
block alignment
when using the IRRUT400 utility 233
block update command 245
BLP (bypass label processing)
for tape volumes 122
browser for workspace data sets 245
buffers
how the RACF manager keeps track of 38
specifying in the data set name table 38

C

cache structure
defining 102
name for 102
RACF support for REBUILDPERCENT 104
reconfiguring 105
cache structure *(continued)*
size of 103
cached auxiliary storage subsystem
using for the RACF database 15
callers of exit routines
summary of 250
case, password
RRSF considerations 147
CDMF algorithm 136, 141
CDT (class descriptor table)
changing an installation-defined class 52
defining new classes 51
deleting an installation-defined class 53
description 50
dynamic 50
generating the table 52
ICHERCDE macro 51
ICHRRCDE module 51
static 50
sysplex communication 50
when the RACF database is shared 50
CFRM policy 102
change count in the ICB
during processing on a shared RACF database 38
checking user authorization 1
CICS
configuring the CICS timeout value range 114
establishing defaults for using RACF 114
TXSeries 115
class
changing 52
defining new classes 51
deleting 53
class descriptor table (CDT)
changing an installation-defined class 52
defining new classes 51
deleting an installation-defined class 53
description 50
dynamic 50
ENF signal 54
generating the table 52
ICHERCDE macro 51
ICHRRCDE module 51
static 50
sysplex communication 50
when the RACF database is shared 50
classroom courses, RACF xv
CLPA parameter
for replacing the started procedures module 109
command direction
path through network 140
command direction, automatic
commands issued from the RACF parameter library 185
command prefix
default 79
specifying 78
command prefix facility (CPF)
failed registration attempt 83
registering a command prefix with 78
commands
directing 128
directing automatically 128
exit routines for 259, 263
issued by exits 148

commands *(continued)*
MVS START command 83
operator commands, RACF 86
RACLINK 127
RESTART 83
running in the RACF subsystem 86
sample exit to fail 268
SET 155, 157
STOP 85
TARGET 159, 191
that do not modify RACF profiles 319
that have recovery routines 320
that perform multiple operations 321
that perform single operations 321
that propagate for RACF sysplex communication 325
to restart the RACF subsystem 83
with sensitive information 115
Commercial Data Masking Facility (CDMF) algorithm 136
completion code
385 286
concepts, basic RACF 2
configuration
RRSF node 343
Configuration Assistant for z/OS Communications Server 135
configuration worksheet, RRSF 343
connections between RRSF nodes
description 132
dormant 133
operative 132
recycling 330
states 133
contact
z/OS 363
control points 105
control statements
for IRRUT100 utility 216, 217
for IRRUT200 utility 223, 225
for IRRUT400 utility 236
control unit
selecting for the RACF database 15
controlling access to resources 1
coordinator in a data sharing group
description 325
COPYAUTH parameter
when using IEHMOVE 118
copying a database
example using IRRUT400 242, 244
to a database with the exact same size 218
to a different device type 233
to a larger database
example using IRRUT400 243
to a larger or smaller database 233
to the same device type 218
using a two-stage process 244
using IRRUT200 218
using IRRUT400 233, 234
coupling facility
defining structures in CFRM policy 102
description 101
failures 315
RACF support for REBUILDPERCENT 104
rebuild support for RACF structures 318
reconfiguring RACF structures 105
coupling facility *(continued)*
sysplex recovery scenarios requiring non sysplex-communication/datasharing mode 319
sysplex recovery scenarios requiring XCF-local mode 318
using with a non-shared database 101
using with a single-system sysplex 101
courses about RACF xv
cross-reference report
produced by IRRUT100 utility 213
CSA
storage requirement 341
Custom Field Validation Exits 303
customization
assigning the data set in the database for a profile 39
changing the RACF report writer options 62
CICS timeout value range 114
data set name table 37
database range table 39
defining resource classes 50
DES (Data Encryption Standard) algorithm for password authentication 56
duplicating updates on backup database 44
enabling sysplex communication 37, 44
enabling sysplex data sharing 37, 44
maintaining statistics on backup database 44
masking algorithm for password authentication 56
number of resident data blocks 38
password authentication algorithm 56
RACF database options 40
RRSF environment 191
specifying ICHRDSNT and ICHRRNG 40
subsystem command prefix 78
using parmlib member 40
CVTSNAME field 160

D

D-B state 134
D-E state 134
D-L state 134
D-R state 134
DASD data set
discrepancies between the profile and the indicator 321, 328
failures during system operations when RACF-protected 328
moving a data set with a discrete profile to a RACF-inactive system 120
moving a RACF-indicated data set to a non-RACF system 120
moving a RACF-indicated data set to a RACF-active system 119
not allowing duplicate profile names 356
operating considerations 116
renaming RACF-protected data sets 117
scratching 122
using access method service commands 121
using IEHMOVE with the ADSP attribute 118
using IEHMOVE with the COPYAUTH parameter 118
using the IMPORT command 121
using the IMPORTRA command 121
using the LISTCAT command 121
using the REPRO command 121

DASD data set *(continued)*
using the RESETCAT command 121
using utilities when RACF-protected 116
using utilities with the group-OPERATIONS attribute 117
using utilities with the OPERATIONS attribute 117
DASD device
selecting for the RACF database 15
DASD volume
moving between systems 122
operating considerations 122
scratching DASD data sets 122
DASDVOL authorization
operating considerations 122
data blocks
how resident blocks affect system performance 17
location of storage 38
size of 38
specifying in ICHSECOP 356
specifying resident blocks in the data set name table 38
data encryption in an RRSF network 136
data masking in an RRSF network 135
data set name table
description 37
example of using 45, 46
format of the flag field 44
selecting the number of resident data blocks 17
specifying the number of resident data blocks 38
specifying the RACF data set names 17
specifying the sysplex communication options 44, 101
data sharing mode
description 101
relationship to RRSF modes 100
database
activating 311
alternate database 6
authenticating the passwords, password phrases, and OIDCARD data 56
backing up 7, 16
backup database 6
commands that modify only one profile 321
commands that perform multiple operations 321
considerations 1
copying
example of using IRRUT400 242–244
example using IRRUT200 224
to a database with a different size 234
to a database with same size 218
to a different device type 234
to same device type 218
using a two-stage process 244
using IRRUT200 utility 218
using IRRUT400 utility 234
creating 10
customizing
range table 39
DASD device 10
deactivating 311
determining percentage of space used 231
DFSMSdss DEFRAG and 12
discrepancies between profiles 319, 321
effect when backup inactive in a sysplex 102
encrypting the passwords, password phrases, and OIDCARD data 56
database *(continued)*
extending with IRRUT400 utility 234
failures during RACF manager processing 327
formatting with IRRMIN00 210
fragmentation 12
identifying inconsistencies using IRRUT200 218
index
scanning blocks 226
statistics from IRRUT200 226
initializing 209
insufficient space 313
insufficient space in 12
levels of backup 17
locating occurrences of a user ID or group name 213
location of 10
locking
example of using IRRUT400 243
maximum number of data sets in 6
merging
example of using IRRUT400 243
modifying records in 245
monitoring usable space in 12
moving 10
multiple data sets 6
overview 3
prompting the operator for the data set name 37
quiescing 311
records initialized for a new database 211
recovery procedures 309
removing references to deleted IDs 244
reorganizing
using IRRUT400 utility 235
repairing
using IRRUT400 utility 235
residual authorities, removing 244
restoring 312
selection of control unit and device 15
shared between systems
application identity mapping (AIM) 87
sharing between systems 7
sharing data between remote systems 9
specifying options
ICHSECOP module 355
splitting
example of using IRRUT400 243
storage requirement
factors affecting 335
formula 335
summary statistics from IRRUT200 231
switching 6, 311
sysplex communication option 9
sysplex data sharing option 9
template level in use 68
templates
overview 3
updating 209
unloading to sequential file 244
unlocking
example of using IRRUT400 244
using resident index and data blocks 18
using the RVARY command 309
using utilities on
IRRIRA00 204
IRRMIN00 209

database *(continued)*
using utilities on *(continued)*
IRRUT100 213
IRRUT200 218
IRRUT400 233
summary 203
when to issue commands that update 19, 20
database configuration
parmlib support 40
database range table
description 39
example of using 48
location 47
when using IRRUT400 utility 237
database unload utility (IRRDBU00) 244
DATASET class
using a global access table for 32
Db2
protecting Db2 data 116
DBSYNC EXEC 147
DD statements (JCL)
for the IRRIRA00 utility 208
for the IRRMIN00 utility 212
ddname
for IRRUT400 input data sets 237
for IRRUT400 output data sets 237
deactivating
automatic direction 156
RACF databases 311
DEF state 134
deferred step restart 124
DEFINE system operation
failures during 328
defined state 134
DEFRAG
using with a RACF database 12
DELDSD command
when moving a data set with a discrete profile 120
when moving a RACF-indicated data set to a non-RACF system 120
DELETE keyword on TARGET command 174
DELETE system operation
failures during 328
DENYINBOUND keyword on TARGET command 160
DES (Data Encryption Standard) algorithm
using 56
DES algorithm 58
DESCRIPTION keyword on TARGET command 161
device
UCB above 16MB 122
DFSMS (Data Facility Storage Management Subsystem)
information in the RACF database 115
DFSMS enhanced data integrity (EDI) 12
DFSMSdss DEFRAG
using with a RACF database 12
diagnostic capability
IRRIRA00 207
IRRMIN00 212
IRRUT100 214
IRRUT200 220
Diagnostic Capability
IRRUT400 236
directed command
path through network 140
disabling RACF 67
discrete profile
changes when the data set is renamed 118
moving a data set with a discrete profile to a RACF-inactive system 120
rules for renaming a data set 117
updating with the correct volume serial number 121
using IEHMOVE with the COPYAUTH parameter 118
dormant by local request state 134
dormant by mutual request state 134
dormant by remote request state 134
dormant connection 133
dormant in error state 134
DORMANT keyword on TARGET command 173
DSMON (data security monitor)
RACF exits report 248
dumps
of the RACF database 7
DUPDATASETS keyword
IRRUT400 utility 240
duplicate data set names
disallowing in ICHSECOP 356
how IRRUT400 handles 240
dynamic allocation parameters
in the ICHRSMFI module 63
dynamic exits facility 263
dynamic parse
automating initialization of 72
determining the level in use 68
RRSF considerations 146
dynamic started procedures table 108

E

ECSA
storage requirement 342
EDI (enhanced data integrity), DFSMS 12
ELSQA
storage requirement 341
enabling RACF 67
encoded map
of a BAM block 230
sample printout by IRRUT200 232
encryption 56
Encryption of data at rest 21
encryption of data in an RRSF network 136
end-of-volume processing
failures during 329
ENF 62 signal 54
ENF 71 signal 55
ENF 79 signal 55
ENF signal 54
enhanced data integrity (EDI), DFSMS 12
ENQ names used by RACF 88
ENVR object
and ICHRFX03 exit 289
and ICHRFX04 exit 295
description 253
EOV processing
failures during 329
EPLPA
storage requirement 341
erase-on-scratch
how it affects system performance 21, 22

ESQA
storage requirement 341
EXEC statement
for IRRIRA00 utility 208
for IRRMIN00 utility 212
PARM parameters when executing IRRUT400 238
exit routine
ACEE compression/expansion 253
availability of started procedure name to 106
commands 263
examining during RACF failures 309
extended addressing 248
for RACF commands 259
how they affect system performance 22
ICHCCX00 259, 262
ICHCNX00 259
ICHDEX01 276
ICHDEX11 276
ICHPWX01 268
ICHPWX11 272
ICHRCX01 281
ICHRCX02 282
ICHRDX01 285
ICHRDX02 286
ICHRFX01 287
ICHRFX02 292
ICHRFX03 288
ICHRFX04 294
ICHRIX01 300
ICHRIX02 301
ICHRLX01 298
ICHRLX02 298
ICHRSMFE 302
ICHRTX00 308
ICHRTX01 308
IRRACX01 253
IRRACX02 253
IRREVX01 263
IRRSXT00 308
naming convention table 249
new password 268
new-password-phrase 272
overview 247
password authentication 276
possible uses of
allowing access when RACF Is inactive 283
controlling access of a shared user ID 284, 296
modifying data set naming conventions 249
password quality control 271
protecting the user's resources from the user 283
RACF exits report from DSMON 248
RACROUTE REQUEST=AUTH
postprocessing 282
preprocessing 281
RACROUTE REQUEST=DEFINE
postprocessing 286
preprocessing 285
RACROUTE REQUEST=FASTAUTH
postprocessing 290, 292, 294
preprocessing 287, 288
RACROUTE REQUEST=LIST
pre- and postprocessing 298
RACROUTE REQUEST=VERIFY(X)
postprocessing 301
exit routine *(continued)*
RACROUTE REQUEST=VERIFY(X) *(continued)*
preprocessing 300
report writer 302
requirements for 247
RRSF considerations 147
SAF callable services router 308
SAF router 308
save area 247
selection 298
summary of callers 250
use of ASXBSENV pointer to ACEE 250
use of PUTLINE 250
use of TCBSENV pointer to ACEE 250
use of TPUT 250
use of WTO 250
exit-generated command 148
exit-generated update 148
exits report
from DSMON 248
extending a database
using IRRUT400 234
extents
when using the IRRUT400 utility 238
EZB.INITSTACK resource in SERVAUTH class 154
EZB.STACKACCESS resource in SERVAUTH class 154

F

failsoft mode
description 101
failsoft processing
description 112
exits called 113
for RACROUTE REQUEST=DEFINE 356
general considerations 113
how it can affect system performance 20
impact on users 114
permanent 113
temporary 113
failures
coupling facility 315
during ALLOCATE or DEFINE operations 328
during EOV operation 329
during RACF command processing 319
during RACF manager processing 327
during RENAME or ALTER operations 329
during SCRATCH or DELETE operations 328
during system operations on data sets 328
failsoft processing 114
RACF parameter library 329
recovery procedures 309
shutting down the RACF subsystem 333
using user ID in SYS1.UADS to logon 309
VSAM 331
feedback xvii
flags
flag field in the data set name table 44
FLPA
storage requirement 341
FORCE command (MVS)
using RACF STOP command instead of 85
formatted output of the index blocks 227
fragmentation in the RACF database 12

FREESPACE keyword
IRRUT400 utility 240
using with IRRUT400 238

G

GDG (generation data group)
renaming individual data sets 117
general resource
definition 49
general resource class
changing 52
changing the class descriptor table 52
defining new classes 51
generic entries
coding in the started procedures table 109, 110
generic profile
considerations when renaming data sets 118
during authorization checking 32
internal name for the range table 39
modified by RACF 39
performance considerations 32
when moving a RACF-indicated data set to a non-RACF system 120
generic profile processing
shared database considerations when disallowed 9
generic profiles
customizing the number of lists of that RACF maintains 34
effect on performance 33
GENERICANCHOR operand on SET command 34
GENLIST processing
effect on system performance 23
GID mapping
improving performance 35
VLF class needed for 74
global access checking
for bypassing normal RACROUTE REQUEST=AUTH processing 32
how it affects system performance 22
when moving a RACF-indicated data set to a non-RACF system 120
global access table
for the DATASET class 32
the entry &RACUID.**/ALTER 32
global resource serialization 87
GLOBALAUDIT operand
RALTER command
effect on system performance 19
group
information about provided by IRRUT100 utility 213
large, effect on performance 36
universal, effect on performance 36
group data set
preventing accidental destruction of data 284
group name
listing all occurrences on the RACF database 213
group profile 2
group tree in storage
activating 86
group-OPERATIONS attribute
determining the owner field when using IEHMOVE 119
when renaming a RACF-indicated data set 118
when using utilities 117
GRPACC (group access) attribute
when renaming a RACF-indicated data set 118
GTF traces 156

I

I/O activity
RVARY command 311
I/O device
UCB above 16MB 122
IBM Commercial Data Masking Facility (CDMF) algorithm 136
ICB (inventory control block)
change count 38
RBAs of the templates defined 229
ICH408I message 122
ICH508I message 249
ICH522I message 110
ICH579E, message 5
ICH702A message 310
ICH703A message 310
ICHAUTAB module
changing 112
ICHCCX00 exit routine
callers of 262
parameter list 262
return codes 262
uses of 259
when entered 259
ICHCNX00 exit routine
callers of 259
parameter fields available to 260
parameter fields that can be changed 260
parameter list 259
return codes 261
uses of 259
when entered 259
ICHDEX01 exit routine 277
ICHDEX11 exit routine 279
ICHERCDE macro
defining new classes in the class descriptor table 51
generating the class descriptor table 52
ICHNCV00 module 249
ICHPWX01 exit routine
conditions for gaining control 268
parameter list 269
return codes 270
ICHPWX11 exit routine 272
ICHRCX01 exit routine
parameter list 281
preventing accidental destruction of data 284
return codes 281
when RACF is inactive 283
ICHRCX02 exit routine
return codes 283
what RACF does before it receives control 282
ICHRDSNT module
description 37, 44
example of using 45, 46
specifying the sysplex communication options 44
ICHRDX01 exit routine
called during failsoft processing 113
requirements for 284
return codes 285

ICHRDX02 exit routine
requirements for 284
return codes 286
ICHRFR00 module
receiving control from the SAF router 105
ICHRFR01 module
description 53
ICHRFR0X module
description 105
ICHRFRTB macro
defining new entries in the router table 53
ICHRFX01 exit routine
environment executed in 287
parameter list 287
requirements for 287
return codes 288
ICHRFX02 exit routine
environment executed in 292
parameter list 292
reason codes 294
requirements for 292
return codes 294
ICHRFX03 exit routine
environment executed in 288
parameter list 288
requirements for 287, 288
return codes 290
ICHRFX04 exit routine
environment executed in 294
parameter list 294
reason codes 296
requirements for 294
return codes 296
ICHRIN03 module
assigning a user ID to the RACF subsystem 81
coding 109
defining started procedures 109
description 109
examples of entries 111, 112
format of the entries
* in the procedure name field 110
= for the user ID or group name 110
generic entries 109
ICHRIX01 exit routine
creating and initializing the ACEE 301
making password checks 271
return codes 301
ICHRIX02 exit routine
return codes 302
what RACF does before it receives control 301
ICHRLX01 exit routine
functions during RACROUTE REQUEST=LIST processing 297
requirements for 297
return codes 298
ICHRLX02 exit routine
functions during RACROUTE REQUEST=LIST processing 297
parameter list 298
requirements for 298
return codes 299
uses for 297
ICHRRCDE module
adding installation-defined classes 51
ICHRRCDE module *(continued)*
changing installation-defined classes 52
deleting installation-defined classes 53
ICHRRNG module
description 47
example of using 48
location 47
relationship to IRRUT400 output data sets 237
table format 47
when using IRRUT400 utility 237
ICHRSMFE exit routine
parameter list 303
return codes 303
uses of 302
when it is called 302
ICHRSMFI module
changing 62
format of 63, 64
ICHRTX00 exit 308
ICHRTX01 exit 308
ICHSECOP module
bypassing RACF initialization processing 355
description of options 355
disallowing duplicate names for DASD data set profiles 356
format 355
resident data blocks, selecting the number of 356
ICHSFR00 module
description 105
ICKDSF (Device Support Facilities) system utility
accessing a RACF-protected DASD data set 116
identifying
RACF users 1
identity context reference, and ICHRIX01 exit 299
identity mapping profiles
recovering from errors 323
IEBCOMPR system utility
accessing a RACF-protected DASD data set 116
IEBCOPY system utility
accessing a RACF-protected DASD data set 116
IEBDG system utility
accessing a RACF-protected DASD data set 116
IEBEDIT system utility
accessing a RACF-protected DASD data set 116
IEBGENER system utility
accessing a RACF-protected DASD data set 116
use by IRRUT200 219
IEBISAM system utility
accessing a RACF-protected DASD data set 116
IEBPTPCH system utility
accessing a RACF-protected DASD data set 116
IEBUPDTE system utility
accessing a RACF-protected DASD data set 116
IEFSSNxx member of SYS1.PARMLIB 78
IEHINITT system utility
restricting its use 123
IEHLIST system utility
accessing a RACF-protected DASD data set 116
IEHMOVE system utility
accessing a RACF-protected DASD data set 116
determining the owner 119
renaming RACF-indicated data sets 117
using with RACF-indicated DASD data sets 118
using with the ADSP attribute 118

IEHMOVE system utility *(continued)*
using with the COPYAUTH parameter 118
IEHPROGM system utility
renaming RACF-indicated data sets 117
IKJTSOxx 72
IMPORT command
using on RACF-protected VSAM data sets 121
IMPORTRA command
using on RACF-protected VSAM data sets 121
IMS (Information Management System)
setting a pointer to the ACEE 123
in-storage profile
using RACROUTE REQUEST=LIST to build 297
inactive (RACF)
allowing access 283
by bypassing RACF initialization processing 355
inactive state for RRSF listener process 132
index blocks
formatted output from IRRUT200 227
sample formatted output from IRRUT200 227
scanning with IRRUT200 utility 226
structure correction with IRRUT400 utility 234
unformatted output by IRRUT200 226
index compression
when using the IRRUT400 utility 233
index entries
problems due to failures during RACF manager processing 327
index structure
correcting when using IRRUT400 utility 234
initial state 134
initialization processing
bypassing 355
initialization routine
loading the RACF exit routines 247
locating the naming convention table module 249
initialization worksheet, RRSF 343
initializing state for RRSF listener process 132
INITSTACK resource in SERVAUTH class 154
INITSTATS option on SETROPTS 29
INMSG data set for RRSF 136
installation exit routine 247
installation-defined class
adding 51
changing 52
deleting 53
installing RACF
formatting the RACF database 210
storage requirements 335
installing the REXX RACVAR function 124
insufficient space condition on the RACF database 313
inventory control block (ICB)
change count 38
IPL
of a RACF data sharing group 49
use of the ICHSECOP module 355
IRR402I message 327
IRR403I message 327
IRR404I message 327
IRRACEE class 32
IRRACX01 and IRRACX02 exit routines 253
IRRADU00 utility 245
IRRBRW00 utility 245, 331
IRRDBU00 utility 244
IRRDPI00 command
authorization 71
automating 72
errors and return codes 71
syntax 68
IRRDPSDS data set
determining the level in use 68
RRSF considerations 146
IRRDPTAB started procedure 72
IRREVX01 exit point 263
IRREVX1A sample exit 268
IRREVX1B sample exit 268
IRRGTS class 86
IRRIRA00 utility
DD statements for 208
description 204
example 208
input 208
output 208
return codes 208
using 208
IRRMIN00 utility
comparison to IRRUT400 utility 238
DD statements for 212
description 209
input 212
output 212
return codes 212
using 212
IRRPRMxx parmlib
sample 44
IRRPRMxx parmlib member
error processing 43
multiple parmlib members 43
sharing a database 43
IRRRID00 utility 244
IRRSEQ00 callable service
requirement for RACF subsystem 76
IRRSMAP VLF class 35, 75
IRRSXT00 exit 308
IRRTEMP2 3
IRRUT100 utility
associated exit routine 214
control 236
description 213
example 217
information provided by the report 213
input and output 216
job control statements 216, 236
sample output of the printed report 217
using 216
utility control statements 217
work data set 215
IRRUT200 utility
BAM/allocation comparison 229
control 223
description 218
example 224
for taking dumps of the RACF database 7
functions 218
identifying problems with the RACF database 327
input and output 222
job control statements 223
return codes 232

IRRUT200 utility *(continued)*
sample output
formatted alias index blocks 229
formatted index blocks 227
of the encoded map 232
unformatted index blocks 226
scanning the index blocks 226
using 221
using a copy to update the RACF database 20
utility control statements 222, 225
IRRUT400
LOCKINPUT keyword 238
IRRUT400 utility
comparison to IRRMIN00 utility 238
copying a database 234
description 233
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 splitting a database using RANGE keyword 242
example of unlocking a database 244
examples of coding 242
executing
input database specification 237
output data set processing 238
output data set selection 238
output database specification 237
parameter specification 238
processing of conflicts and inconsistencies 240
extending a database 234
how it works 233
parameters 238
RANGEDD statement 236
reorganizing a database 235
repairing a database 235
return codes 241
when not to use 240
IRRVERLD program
description 125
return codes 125
ISMF
use of panel driver interface 124
ISPF
issuing commands from 115
ISPLOG data set
logging commands with sensitive information 115

J

JCL (job control language)
examples of coding IRRUT400 utility 242
EXEC statement for IRRIRA00 208
EXEC statement for IRRMIN00 212
for creating a database 10
for IRRUT100 utility 216
for IRRUT200 utility 223
for IRRUT400 utility 236
parameters ignored when bypassing RACF initialization 355
specifying bypass label processing 122
to activate the RACF subsystem 82
JES initialization
specifying bypass label processing 122
JOB statement (JCL)
for started procedures 106
parameters ignored when bypassing RACF initialization 355
specifying the password when restarting jobs 124
jobs
restarting 124

K

KDFAES
algorithm 57
KDFAES is active
ICHDEX01 277
ICHDEX11 277
KDFAES not active
ICHDEX01 276
ICHDEX11 276
key qualifiers for generic profiles 33
keyboard
navigation 363
PF keys 363
shortcut keys 363
KEYQUAL value in class descriptor table 33

L

LABEL parameter on DD statement
specifying bypass label processing 122
large group, effect on performance 36
large profile, effect on performance 35
LIST keyword on TARGET command 163
LISTCAT command
using on RACF-protected VSAM data sets 121
listener process for RRSF
default port number 135
port, protecting 153
listing the attributes of an RRSF node 155
LISTPROTOCOL keyword on TARGET command 163
LOCAL keyword on TARGET command 161
local mode for an RRSF node
configuration example 194
description 131
relationship to sysplex communication modes 100
local node 129
local peer system 130
local system 130
location of resident data blocks 356
location of the RACF database 10
locking a database
example using IRRUT400 243
LOCKINPUT keyword
IRRUT400 utility 238
using with IRRUT400 234
logging
how it affects system performance 18
using RACROUTE REQUEST=AUTH exit routine to modify 281
logging and reporting 2
LOGREC records for SETROPTS processing on a sysplex 327
LSQA

LSQA *(continued)*
storage requirement 341
LU name
how to determine 161
LUNAME keyword on TARGET command 161

M

macros issued by exits 148
MAIN keyword on TARGET command 161
MAIN switch procedure
mixed-release network 359, 360
main system
configuring a new one 175
defining 161
description 129
selecting 161
maintenance
restarting a function after applying 85
mapping UIDs and GIDs
improving performance 35
VLF classes needed for 74
masking algorithm 56, 276
masking of data in an RRSF network 135
member systems 130
merging data sets
example using IRRUT400 243
merging data sets in the database
using IRRUT400 233
message ICH579E 5
messages
ICH408I 122
ICH508I 249
ICH522I 110
IRR402I 327
IRR403I 327
IRR404I 327
mixed case password
RRSF considerations 147
mixed protocols in an RRSF network 135
mode
data sharing 101
failsoft 101
local 131
non–data sharing 100
read-only 101
remote 131
MODENAME keyword on TARGET command 162
moving a multivolume RACF-indicated DASD data set 120
moving a RACF-indicated DASD data set
between systems 119
to a non-RACF system with RACF indicator checking 120
to a RACF-active system 119
with a discrete profile to a RACF-inactive system 120
moving DASD volumes between systems 122
moving tape volumes
between systems 123
multivolume tape data sets 123
multiple input data sets
considerations when using IRRUT400 utility 234
multiple users per address space 123
multisystem node 129
multisystem RRSF node
adding a system to 174
multisystem RRSF node *(continued)*
configuring a new main system 175
deleting a system from 174
selecting the main system for 161
system requirements 145
multivolume tape data set 123
MVS router 308
MVS START command 83

N

naming convention table
functions it should perform 249
RRSF considerations 147
use of 249
when processing occurs 249
naming conventions
changing the standard naming conventions 249
for RRSF workspace data sets 136
modifying with exits 249
when defining DASD data set profiles 116
navigation
keyboard 363
NETSTAT command 154
network
RRSF 128
network protocol for a connection
changing 181
network-qualified name
and workspace data set name 137
on TARGET command 162
new MAIN
downlevel 359
uplevel 360
new-password exit routine 268
NOALIGN keyword
IRRUT400 utility 240
NOCMDVIOL operand
SETROPTS command
effect on system performance 19
NODE keyword on TARGET command 160
nodes, RRSF
breaking a connection with 174
changing the protocol for connections 181
connection states 133
connections between 132
defining 159
description 129
dormant connection 133
local 129
local mode 131
mismatches in definitions of 160
multisystem 129
operative connection 132
protocol conversion 181
recycling connections 330
remote 129
remote mode 131
single-system 129
NODUPDATASETS keyword
IRRUT400 utility 240
NOFREESPACE keyword
IRRUT400 utility 240
NOLOCKINPUT keyword

NOLOCKINPUT keyword *(continued)*
IRRUT400 utility 238
non-shared RACF database
consideration when changing to shared 7
processing of in-storage buffers 38
non-VSAM data set
failures during ALLOCATE operation 328
failures during EOV system operation 329
failures during RENAME operation 329
failures during SCRATCH operation 328
formatting for use as a RACF database 210
turning off the RACF indicator and deleting the profile 120
turning off the RACF indicator and preserving the profile 120
when extended on one system and moved to another 120
non–data sharing mode
description 100
relationship to RRSF modes 100
nonmain system 129
NOSAUDIT operand
SETROPTS command
effect on system performance 19
NOSET operand
ADDSD command
when moving a RACF-indicated data set 119
not defined state 134
NOTABLE keyword
IRRUT400 utility 239

O

O-A state 133
O-E state 134
O-P-C state 133
O-P-V state 133
OIDCARD data
authenticating algorithms 56
checking the validity with RACROUTE REQUEST=VERIFY 299
old MAIN
downlevel 359
uplevel 360
old MAIN and new MAIN
downlevel 359
uplevel 360
OPEN macro instruction
RACF authorization checking 116, 123
operating considerations
commands issued from ISPF 115
DASD data sets 116
for DASD volumes 122
moving a data set with a discrete profile to a RACF-inactive system 120
moving a multivolume RACF-indicated data set between systems 120
moving a RACF-indicated DASD data set between systems 119
moving a RACF-indicated DASD data set to a non-RACF system 120
moving a RACF-indicated DASD data set to a RACF-active system 119
moving DASD volumes between systems 122
operating considerations *(continued)*
moving tape volumes between systems 123
multiple users per address space 123
protecting Db2 data 116
RACF panel driver interface 124
renaming RACF-protected data sets 117
restarting jobs 124
REXX RACVAR function 124
scratching DASD data sets 122
tape volumes
bypass label processing 122
protection for unlabeled tapes 123
TSO profiles in the RACF database 115
UCBs above 16MB 122
using access method service commands
IMPORT command 121
IMPORTRA command 121
LISTCAT command 121
REPRO command 121
RESETCAT command 121
using IEHMOVE with the ADSP attribute 118
using IEHMOVE with the COPYAUTH parameter 118
using utilities on RACF-protected DASD data sets 116
using utilities on RACF-protected tape volumes 123
using utilities with the group-OPERATIONS attribute 117
using utilities with the OPERATIONS attribute 117
operating system and RACF interaction 2
OPERATIONS attribute
determining the owner field when using IEHMOVE 119
when RACF is inactive 283
when renaming a RACF-indicated data set 118
when using utilities 117
operative active state 133
operative connection 132
operative in error state 134
OPERATIVE keyword on TARGET command 172, 173
operative pending connection state 133
operative pending verification state 133
operator commands, RACF 86
operator prompts
during failsoft processing 113
for an asterisk in the data set name table 37
options
changing the ICHAUTAB module 112, 356
changing the ICHRSMFI module 62
changing the RACF report writer options 62
CICS timeout value range 114
data set name table 37
database range table 39
defining resource classes 50
DES (Data Encryption Standard) algorithm for password authentication 56
duplicating updates on backup database 44
enabling sysplex communication 44
enabling sysplex data sharing 44
ICHDEX01 exit 56
maintaining statistics on backup database 44
masking algorithm for password authentication 56
number of resident data blocks 38
password authentication algorithm 56
RRSF environment 191
specifying RACF database options
database range table 39

options *(continued)*
specifying RACF database options *(continued)*
ICHSECOP module 355
using parmlib member 40
subsystem command prefix 78
using the system authorization facility (SAF) 105
which data set to place each profile on 39
OUTMSG data set for RRSF 136
overview
RACF database options 37
OWNER field
resolving conflicts with RACROUTE REQUEST=LIST
selection exit routine 298
owner of the profile
when using IEHMOVE with the COPYAUTH parameter 119

P

panel driver interface
operating considerations 124
parameter library, RACF
attributes 185
automatically processing at initialization 186
blank lines
comments 186
commands that can be issued from 185
configuring RRSF without 185
continuing commands in 186
description 184
initializing dynamic parse from 72
member names 185
order of commands in 191
processing using SET INCLUDE 188
recovering from errors 329
running IRRDPI00 from 72
security for 184
sharing 190
parameter lists
ICHCCX00 exit routine 262
ICHCNX00 exit routine 259
ICHPWX01 exit routine 269
ICHRCX01 exit routine 281
ICHRFX01 exit routine 287
ICHRFX02 exit routine 292
ICHRFX03 exit routine 288
ICHRFX04 exit routine 294
ICHRLX02 exit routine 298
ICHRSMFE exit routine 303
modifying with the SAF router exit routine 308
PARM field of the EXEC statement
parameters when executing IRRUT400 238
PassTicket
validating 58
password
authenticating algorithms
DES (Data Encryption Standard) 56, 276
installation-provided 276
masking 56, 276
two-step method 58
authentication exit routines 276
checking validity with RACROUTE REQUEST=VERIFY 299
encryption 56
password *(continued)*
envelope, requirement for RACF subsystem 76
for activating or deactivating RACF 310
for changing the RACF operating mode 310
for RVARY 310
for switching the RACF database 310
new-password exit routine 268
PassTicket as an alternative for 58
quality control 271
synchronizing 128
use by RACF 1
when restarting jobs 124
PASSWORD command
invoking the new-password exit routine 268
making password checks 271
password phrase
authenticating algorithms 56
envelope, requirement for RACF subsystem 76
new-password-phrase exit routine 272
synchronizing 128
password rules
RRSF considerations 146
password, automatic direction of updates 128
password, mixed case
RRSF considerations 147
performance
effect of large groups on 36
effect of large profiles on 35
effect of universal groups on 36
effect of UNIXMAP class 35
factors affecting the system 15
generic profiles 33
how erase-on-scratch can affect 21, 22
how exit routines affect 22
how global access checking affects 22
how logging affects 18
how RACF commands can affect 19
how RACROUTE REQUEST=AUTH processing affects 32
how RACROUTE REQUEST=FASTAUTH processing affects 32
how RACROUTE REQUEST=VERIFY processing affects 31
how SETROPTS GENLIST processing affects 23
how SETROPTS RACLIST processing affects 23
how statistics gathering affects 29
how UID and GID mapping affects 35
how utility programs affect 20
using resident index and data blocks 17
VLF considerations 35
z/OS UNIX System Services applications 35
permanent failsoft 113
persistent verification
requirement for RACF subsystem 76
PLPA
storage requirement 341
policy, CFRM 102
port number , default for listener socket in an RRSF network 135
PORTNUM keyword on TARGET command 162
POSIT number
effect on SETROPTS STATISTICS 30
precaution when changing 52
precaution when deleting a class 53
postprocessing exit routine

postprocessing exit routine *(continued)*
ICHRCX02 282
ICHRDX02 286
ICHRIX02 301
RACROUTE REQUEST=AUTH
return codes 283
RACROUTE REQUEST=DEFINE
requirements for 284
return codes 286
RACROUTE REQUEST=FASTAUTH
environment executed in 292, 294
parameter list 292, 294
reason codes 294, 296
requirements for 292, 294
return codes 294, 296
RACROUTE REQUEST=LIST
requirements for 297
return codes 298
RACROUTE REQUEST=VERIFY
return codes 302
RACROUTE REQUEST=VERIFY(X) 301
prefix
specifying for operator commands 78
PREFIX keyword on TARGET command 162
preprocessing exit routine
how system performance is affected 22
ICHCCX00
parameter list 262
return codes 262
ICHCNX00
calling before IRRUT100 utility 214
parameter list 259
return codes 261
ICHRDX01 285
ICHRIX01 300
RACROUTE REQUEST=AUTH
parameter list 281
return codes 281
RACROUTE REQUEST=DEFINE
requirements for 284
return codes 285
RACROUTE REQUEST=FASTAUTH
environment executed in 287, 288
parameter list 287, 288
requirements for 287, 288
return codes 288, 290
RACROUTE REQUEST=LIST
requirements for 297
return codes 298
RACROUTE REQUEST=VERIFY
return codes 301
RACROUTE REQUEST=VERIFY(X) 300
what RACF does before exit receives control 247
privileged attribute
for started procedures 106
profile
commands that do not modify 319
discrepancies between 321
discrepancies with indicator for DASD data sets 328
large, effect on performance 35
not allowing duplicate DASD data set names 356
specifying in ICHSECOP 356
PROTECT line operator in ISMF
use of panel driver interface 124
PROTECT parameter (JCL DD statement)
when restarting jobs 124
protected user ID 81, 106
Protecting data during its life cycle 21
protocol conversion, for RRSF connections 181
protocol for a connection
changing 181
protocol instance 135
PROTOCOL keyword on TARGET command 161
protocols, mixed, in an RRSF network 135
PURGE keyword on TARGET command 173
PUTLINE 250

Q

quiescing RACF database I/O activity 311

R

R_admin callable service
requirement for RACF subsystem 76
RACF
disabling 67
enabling 67
encryption algorithms 56
KDFAES and DES 56
operating considerations 67
performance considerations 15, 31
recovery procedures 114
utilities
IRRIRA00 204
IRRMIN00 209
IRRUT100 213
IRRUT200 218
IRRUT400 233
summary 203
RACF authorized-caller table 112
RACF commands
exit routines for
ICHCCX00 262
ICHCNX00 259
failures during RACF command processing 319
how they can affect system performance 19
operator commands 86
RACLINK 127
RESTART 83
running in the RACF subsystem 86
SET 155
STOP 85
TARGET 159
that do not modify RACF profiles 319
that have recovery routines 320
that perform multiple operations 321
that perform single operations 321
that propagate for RACF sysplex communication 325
RACF database configuration
data set name table 37
RACF database options
overview 37
RACF exits report
from DSMON 248
RACF indicator
turning off 120

RACF manager
failures during processing 327
processing of in-storage buffers 38
return code of 60 39
RACF options 37
RACF parameter library
attributes 185
automatically processing at initialization 186
commands that can be issued from 185
configuring RRSF without 185
continuing commands in 186
description 184
initializing dynamic parse from 72
member names 185
order of commands in 191
processing using SET INCLUDE 188
recovering from errors 329
running IRRDPI00 from 72
security for 184
sharing 190
RACF PROC
sample 82
RACF report writer
default values in the ICHRSMFI module 62
description 63
exit routine 302
format of the ICHRSMFI module 63, 64
list of functions 62
options in the ICHRSMFI module 63
RACF router
receiving control from the SAF router 105
RACF router table
description 105
RACF subsystem
activating 77
assigning a user ID to 80
description 75
multiple 77
parameter library problems 329
recovery procedures 329
recovery when a task stops 330
restarting 82
restarting a function in 83
running commands in 86
sample JCL to activate 82
shutting down 333
specifying the command prefix for 78
RACF subsystem address space
stopping 85
RACF-indicated DASD data sets
moving a multivolume data set between systems 120
moving between systems 119
moving to a non-RACF system with RACF indicator checking 120
moving to a RACF-active system 119
RACF-protected indicator
discrepancies with DASD data set profiles 321, 328
RACFDS address space 101
RACFICE reporting tool 245
RACFRCVY (RACF recovery) started procedure 312
RACFRW utility 245
RACGLIST class
effect on system performance 23
shared database considerations 9
RACLINK command 127
RACLIST processing
effect on system performance 23
RACROUTE macro
invoking the SAF router 308
RACROUTE REQUEST=AUTH
effect on system performance 32
exit routines
uses for 281
preprocessing routine called during failsoft 113
privileged attribute for started procedures 106
trusted attribute for started procedures 106
RACROUTE REQUEST=DEFINE
exit routines 284–286
preprocessing routine called during failsoft 113
when RACF initialization is bypassed 356
RACROUTE REQUEST=FASTAUTH
exit routines 286, 288, 292, 294
use of resident profiles 32
RACROUTE REQUEST=LIST
authorizing use of via RACF authorized-caller table 356
description of processing 297
exit routines 298
RACROUTE REQUEST=VERIFY
authorizing use of via RACF authorized-caller table 356
building the default ACEE 299
effect of on system performance 31
RACROUTE REQUEST=VERIFY CREATE
for multiple users per address space 123
RACROUTE REQUEST=VERIFY(X)
exit routine 299
exit routines 299
RACVAR function for REXX execs
information it provides 124
installing 124
RANGE keyword
IRRUT400 utility 239
range table
example of using 48
location 47
when using IRRUT400 utility 237
range table for ACEE compression and expansion 254
RBA (relative byte address)
of a BAM block 230
of the templates defined in the ICB 229
read-only mode
description 101
relationship to RRSF modes 100
running IRRUT400 in 238
reason codes
from ICHRFX02 exit routine 294
from ICHRFX04 exit routine 296
rebuild of RACF cache structures 105
rebuild support for RACF structures 318
REBUILDPERCENT 104
recovery procedures
for coupling facility failures 315
for failures during ALLOCATE or DEFINE operations 328
for failures during EOV operation 329
for failures during RACF command processing 319
for failures during RACF manager processing 327
for failures during RENAME or ALTER operations 329
for failures during SCRATCH or DELETE operations 328
for failures during system operations on data sets 328

recovery procedures *(continued)*
for hung connection 330
for RACF parameter library problems 329
for the RACF database 309
RACFRCVY started procedure 312
sample procedures 313
shutting down the RACF subsystem 333
synchronization considerations 312
TSO considerations 309
using UADS user ID 309
VSAM failures on the workspace data sets 331
when a task stops 330
when the workspace data sets fill up 331
recovery routines
commands that have 320
recycling a connection 330
REFRESH GENERIC operands
SETROPTS command 28
REFRESH RACLIST operands
SETROPTS command 27
remote mode for an RRSF node
configuration example using APPC/MVS 195
configuration example using TCP/IP 196
description 131
relationship to sysplex communication modes 100
remote node 129
remove ID utility (IRRRID00) 244
RENAME command, TSO
renaming RACF-indicated data sets 117
RENAME on a MOVE statement
when using IEHMOVE 118
RENAME system operation
failures during 329
renaming RACF-indicated data sets
individual data sets of a GDG 117
reorganizing a database
using IRRUT400 235
repairing a database
using IRRUT400 235
replaceable modules
ICHAUTAB 356
ICHRDSNT 37, 44
ICHRIN03 109
ICHRRNG 47
ICHRSMFI 63
ICHSECOP 355
report
produced by IRRUT100 utility 213
sample output from IRRUT100 217
report writer
default values in the ICHRSMFI module 62
description 63
exit routine 302
format of the ICHRSMFI module 63, 64
list of functions 62
options in the ICHRSMFI module 63
REPRO command
using on RACF-protected VSAM data sets 121
RESETCAT command
using on RACF-protected VSAM data sets 121
RESETDENYINBOUNDCOUNT keyword on TARGET command 161
resident data blocks
how they affect system performance 17
resident data blocks *(continued)*
specifying in ICHSECOP 356
specifying in the data set name table 38
resident index blocks
how they affect system performance 17
resident profiles
use by RACROUTE REQUEST=FASTAUTH for authorization checking 32
using RACROUTE REQUEST=LIST to build 297
ways used by RACF 282
resource class
changing 52
class descriptor table 50
defining new classes 51
deleting 53
resource groups
during RACROUTE REQUEST=LIST processing 297
resource managers
using the RACROUTE REQUEST=AUTH 281
RESTART command
using after applying maintenance 85
using to recover from failures 85
restarting
functions in the RACF subsystem 83
jobs 124
the RACF subsystem 82
return codes
60 from the RACF manager 39
from ICHCCX00 exit routine 262
from ICHCNX00 exit routine 261
from ICHPWX01 exit routine 270
from ICHRCX01 exit routine 281
from ICHRCX02 exit routine 283
from ICHRDX01 exit routine 285
from ICHRDX02 exit routine 286
from ICHRFX01 exit routine 288
from ICHRFX02 exit routine 294
from ICHRFX03 exit routine 290
from ICHRFX04 exit routine 296
from ICHRIX01 exit routine 301
from ICHRIX02 exit routine 302
from ICHRLX01 exit routine 298
from ICHRLX02 exit routine 299
from ICHRSMFE exit routine 303
from IRRIRA00 utility 208
from IRRMIN00 utility 212
from IRRUT200 utility 232
from IRRUT400 utility 241
from IRRVERLD program 125
REXX RACVAR function
information it provides 124
installing 124
router table
defining new entries 53
description 53
RRSF (RACF remote sharing facility)
&RACLNDE, using 194
APPC considerations 149
associations between user IDs 127
automatic direction
activating and deactivating 156
breaking a connection with a node 174
changing the protocol for connections 181
configuration examples 194

RRSF (RACF remote sharing facility) *(continued)*
configuration worksheet 343
configuring an RRSF network 154, 191
configuring without the RACF parameter library 185
connections between nodes
description 132
recycling 330
states 133
considerations for new-password exit 269
considerations for new-password-phrase exit 272
controlling
incoming requests 173
outgoing requests 172
customizing 191
data masking 135
defined state 134
defining nodes 159
defining standard sequences of configuration commands 184
directed command
path through network 140
dormant by local request state 134
dormant by mutual request state 134
dormant by remote request state 134
dormant connection 133
dormant in error state 134
dynamic parse version considerations 146
encryption 135, 136
examples of configuration 194
exit considerations 147
initial state 134
initialization scenario 344
initialization worksheet 343
INMSG data set 136
installation exit considerations 147
installation-provided code considerations 148
introduction 9
IRRBRW00 245
IRRDPSDS 146
JES security considerations 194
listener process
default port number 135
description 132
protecting 153
listing attributes of target nodes 163
listing the attributes of an RRSF node 155
local mode 131
local node 129
local peer system 130
local system 130
member systems 130
mixed case passwords 147
multisystem node 129
naming convention table considerations 147
network 128
nodes
defining 159
description 129
local 129
mismatches in definitions of 160
multisystem 129
remote 129
single-system 129
not defined state 134
RRSF (RACF remote sharing facility) *(continued)*
operative active state 133
operative connection 132
operative in error state 134
operative pending connection state 133
operative pending verification state 133
OUTMSG data set 136
parameter library 184
password rule considerations 146
prerequisites 145
protocol conversion 181
protocol instance 135
purging workspace data sets 173
RACF parameter library 184
RACF subsystem address space considerations 149
RACLINK command 127
recovery from parameter library problems 329
recovery when a task stops 330
remote mode 131
remote node 129
RRSFDATA class 191, 193
sample AT-TLS policy 135
security for 193
SET command 155
SETROPTS options considerations 146, 147
shared database considerations 7
single-system node 129
states of connection 133
STOP command considerations 86
subsystem address space considerations 149
TARGET command 159
target nodes, defining 159
TCP/IP 135
template version considerations 145
tracing 156
user ID associations 127
VSAM file browser 245
VTAM considerations 149
worksheet 343
workspace data sets
defining 138
description 136
determining how full 139
increasing the size of 331
maintaining 139
naming conventions 136
preallocating 138
recovering when they fill up 331
size guidelines 139
viewing 331
RRSF node 343
RRSFDATA class
customizing the RRSF environment with 191
establishing security for the RRSF environment with 193
RRSFLIST data set 157
RVARY command
ACTIVE operand 311
failures when propagating 325
I/O activity 311
INACTIVE operand 311
password for 310
SWITCH operand 311
using with shared RACF database 310
RVARYPW operand on SETROPTS command 310

S

SAF (system authorization facility)
  callable services router exit 308
  invoking the MVS router 105
  invoking the SAF router 308
  using 105
SAF router
  description 105
  exit routine 308
save area for installation exits 247
scanning index blocks
  formatted printout of from IRRUT200 227
  unformatted printout of by IRRUT200 226
  with IRRUT200 utility 226
SCRATCH system operation
  failures during 328
scratching DASD data sets 122
security
  administering 2
security administrator
  role of 1
security requirements
  how RACF meets 1
Security Server (RACF)
  disabling 67
  enabling 67
security topics for RACF
  classroom courses xv
security-sensitive fields in a user's profile 74
sending to IBM
  reader comments xvii
serialized access to shared databases 87
serializing access to resources 87
SET command
  customizing the number of lists of generic profiles that RACF maintains 34
  GENERICANCHOR operand 34
  INCLUDE keyword 188
SETROPTS command
  failures when propagating 326
  REFRESH GENERIC operands 28
  REFRESH RACLIST operand 27
  specifying rules for passwords 271
SETROPTS GENLIST processing
  effect on system performance 23
SETROPTS options
  RRSF considerations 146
SETROPTS RACLIST processing
  effect on system performance 23
shared RACF database
  application identity mapping (AIM) 87
  caution for class descriptor tables 50
  classes that do not allow generic profile processing 9
  consideration when changing to non-shared 7
  considerations 87
  in an RRSF network 129
  processing of in-storage buffers 38
  RACGLIST class, considerations 9
  sharing between z/OS and z/VM 8
  specifying the resident data block option 38
  sysplex communication 91, 105
  sysplex communication option 9
  sysplex data sharing option 9
shared RACF database *(continued)*
  using the RVARY command 310
shared RACF parameter library 190
shared user ID
  controlling access 284, 296
shortcut keys 363
signal
  ENF 62 54
  ENF 71 55
  ENF 79 55
signal, ENF 54
signed programs
  verification by RACF 125
single-system node 129
SMF data
  when restoring a RACF database 312
SMF data unload utility 245
SMF records
  when ICHRSMFE exit routine is called 302
SORT/MERGE parameters
  in the ICHRSMFI module 63
space used in RACF database
  determining 231
SPECIAL authority
  sample exit to limit 268
splitting a database
  example using IRRUT400 243
  using IRRUT400 233
splitting a database using RANGE keyword
  example using IRRUT400 242
SPZAP service aid
  adding entries to ICHAUTAB 357
SQA
  storage requirement 341
stack access control for TCP/IP 154
STACKACCESS resource in SERVAUTH class 154
standard naming convention for data set profiles 116
START command, MVS 83
STARTED class
  assigning a user ID to the RACF subsystem 81
  using 108
started procedure
  description 105
  initializing dynamic parse from 72
  privileged attribute 106
  running IRRDPI00 from 72
  trusted attribute 106
started procedures table
  assigning a user ID to the RACF subsystem 81
  coding 109
  defining started procedures 109
  description 109
  examples of entries 111, 112
  generic entries 109
states of connection for RRSF nodes 133
statistics
  from IRRUT200 about the database index 226
  from IRRUT200 about the RACF database 231
  how they affect system performance 29
  on the RACF backup database 17
STATISTICS option on SETROPTS 30
STOP command 85
stopping the RACF subsystem address space 85
storage requirement

storage requirement *(continued)*
coupling facility cache structure 103
RACF database 335
virtual for RACF 341, 342
structure, cache
defining 102
name for 102
RACF support for REBUILDPERCENT 104
reconfiguring 105
size of 103
subsystem address space, RACF
stopping 85
subsystem, RACF
activating 77
assigning a user ID to 80
description 75
multiple 77
parameter library problems 329
recovery procedures 329
recovery when a task stops 330
restarting 82
restarting a function 83
running commands in 86
sample JCL to activate 82
shutting down 333
specifying the command prefix for 78
stopping 85
summary of changes for z/OS version 2 release 3 (v2r3) xix
summary of changes for z/OS version 2 release 4 (v2r4) xix
SWITCH operand
RVARY command 311
switching primary and backup databases 309, 311
symptom records for SETROPTS processing on a sysplex 327
synchronization
of database profiles
establishing 147
of passwords and password phrases 128
when restoring a database 312
when using RVARY when the RACF database is shared 310
SYS1.LOGREC records for SETROPTS processing on a sysplex 327
SYS1.LPALIB
range table 47
SYS1.PARMLIB
IEFSSNxx member 78
SYSNAME keyword on TARGET command 160
sysplex communication
address space 101
cache structure 102
class descriptor table 50
command propagation 50, 325
coordinator 325
coupling facility 102
data sharing mode 101
effect of inactive backup data sets 102
enabling 101
failsoft mode 101
failsoft processing 20, 113
failures when propagating RVARY command 325
failures when propagating SETROPTS command 326
non–data sharing mode 100
option 9
sysplex communication *(continued)*
overview 91, 105
propagation of RVARY commands 113
re-IPLing a RACF data sharing group 49
read-only mode 101
RVARY SWITCH command 311
specifying in the data set name table 44, 101
sysplex data sharing
cache structure 102
coupling facility 102
coupling facility structure definition size 38
data sharing mode 101
defining structures in CFRM policy 102
failsoft processing 113
option 9
RACF support for REBUILDPERCENT 104
RACF support of rebuild interface 105
reconfiguring RACF structures 105
serialized access to shared databases 87
specifying in the data set name table 44, 101
sysplex recovery scenarios requiring non sysplex-communication/datasharing mode 319
sysplex recovery scenarios requiring XCF-local mode 318
SYSPRINT ddname
for IRRIRA00 utility 208
for IRRMIN00 utility 212
SYSRACF
for IRRMIN00 utility 212
system
main
configuring a new one 175
description 129
selecting 161
nonmain
description 129
system generation
specifying bypass label processing 122
system name, identifying for a multisystem RRSF node 160
system operations
failures during operations on RACF-protected data sets 328
system performance
factors affecting 15
how exit routines affect 22
how failsoft processing can affect 20
how global access checking affects 22
how logging affects 18
how RACF commands can affect 19
how RACROUTE REQUEST=AUTH processing affects 32
how RACROUTE REQUEST=FASTAUTH processing affects 32
how RACROUTE REQUEST=VERIFY processing affects 31
how SETROPTS GENLIST processing affects 23
how SETROPTS RACLIST processing affects 23
how statistics gathering affects 29
how utility programs affect 20
using erase-on-scratch 21, 22
using resident index and data blocks 17
system prerequisites for RRSF 145
system programmer
role of 1
system utilities
RACF authorization checking during OPEN 116, 123

T

TABLE keyword
  IRRUT400 utility 239
tape labels
  unlabeled tapes 123
tape volume protection
  and bypass label processing 122
  for unlabeled tapes 123
  when moving tape volumes between systems 123
tape volume sets
  conflicts detected by IRRUT400 utility 240
tape volumes
  moving between systems 123
  moving multivolume tape data sets between systems 123
  using RACF to protect 122
  using utilities when RACF-protected 123
TAPEVOL class
  caution for the range table 39
TARGET command
  ADDRESS keyword 162
  ALLOWINBOUND keyword 160
  DELETE keyword 174
  DENYINBOUND keyword 160
  DESCRIPTION keyword 161
  DORMANT keyword 173
  LIST keyword 163
  LISTPROTOCOL keyword 163
  LOCAL keyword 161
  LUNAME keyword 161
  MAIN keyword 161
  MODENAME keyword 162
  NODE keyword 160
  OPERATIVE keyword 172, 173
  order in which to issue 191
  PORTNUM keyword 162
  PREFIX keyword 162
  PROTOCOL keyword 161
  PURGE keyword 173
  RESETDENYINBOUNDCOUNT keyword 161
  SYSNAME keyword 160
  TPNAME keyword 162
  WORKSPACE keyword 163
target nodes
  defining 157, 159
tasks
  configuring a multisystem node
    steps for 197
  protocol for a connection, changing
    steps 181
  synchronizing database templates
    step for 5
    steps for 4, 5
TCBSENV pointer to ACEE 250
TCP/IP
  default port number for listener socket in an RRSF network 135
  using in an RRSF network 135
templates
  determining the level in use 68
  overview 3
  RRSF considerations 145
  updating 209
templates *(continued)*
  verification by IRRUT200 utility 229
temporary failsoft 113
terminal monitor program
  running IRRUT200 under TMP 222
TME 10 User Administration
  requirement for RACF subsystem 76
TMP (terminal monitor program)
  running IRRUT200 under TMP 222
TPNAME keyword on TARGET command 162
TPUT 250
tracing APPC, IMAGE, and RRSF events 156
tracing TCP/IP events 156
trademarks 370
trusted attribute
  for started procedures 106
TSO considerations
  storing TSO profiles in the RACF database 115
  using user ID in SYS1.UADS to logon 309
TSO information in the RACF database
  operating considerations 115
TSO segment
  using field level access checking to protect 115
two-step method of password authentication 58
TXSeries 115

U

UAUDIT operand
  ALTUSER command
    effect on system performance 19
UCBs above 16MB 122
UID mapping
  improving performance 35
  VLF class needed for 74
UL tapes 123
undefined user
  supplying a user ID 299
unit control blocks (UCBs) above 16MB 122
universal group, effect on performance 36
UNIXMAP class
  how it improves performance 35
unlabeled tapes 123
unlocking a database
  example using IRRUT400 244
UNLOCKINPUT keyword
  IRRUT400 utility 238
user data set
  preventing accidental destruction of data 284
user ID
  assigning to RACF subsystem 80
  listing all occurrences on the RACF database 213
  protected 81, 106
  used by RACF 1
user ID association
  overview 127
user interface
  ISPF 363
  TSO/E 363
user private, above 2G
  storage requirement 342
user private, below 16MB
  storage requirement 342
user security packet 35, 75

user, undefined
  supplying a user ID 299
users
  information on provided by IRRUT100 utility 214
USP 35, 75
utilities
  database cross reference 213
  database initialization 209
  database reorganization 204
  database split/merge/extend 233
  database unload 244
  database verification 218
  for use on the RACF database 203
  how they can affect system performance 20
  IRRADU00 245
  IRRBRW00 245
  IRRDBU00 244
  IRRIRA00 204
  IRRMIN00 209
  IRRRID00 244
  IRRUT100 213
  IRRUT200 218
  IRRUT400 233
  RACF report writer 245
  RACFICE reporting tool 245
  RACFRW 245
  remove ID 244
  RRSF VSAM file browser 245
  SMF data unload 245
  using IEHMOVE with the ADSP attribute 118
  using IEHMOVE with the COPYAUTH parameter 118
  using on RACF-protected DASD data sets 116
  using on RACF-protected tape volumes 123
  using with the group-OPERATIONS attribute 117
  using with the OPERATIONS attribute 117
utilities, system
  RACF authorization checking during OPEN 116
utility control statements
  for IRRUT100 utility 217
  for IRRUT200 utility 225

V

V2R1 deleted information xxii
V2R2 changed information xx, xxi
V2R2 deleted information xx
V2R2 new information xx
VALREXX
  input 307
  output 308
verification of signed programs
  IRRVERLD program 125
verifying RACF users 1
virtual storage requirement for RACF 341, 342
VLF
  GID mapping, class needed for 74
  IRRACEE class 32, 73
  IRRGMAP class, defining 74
  IRRGTS class 86
  IRRSMAP class, defining 75
  IRRUMAP class, defining 74
  removing information from 74
  UID mapping, class needed for 74
  using to improve performance 32, 35
VSAM data set
  failures during ALTER operation 329
  failures during DEFINE operation 328
  failures during DELETE operation 328
  IMPORT command 121
  IMPORTRA command 121
  LISTCAT command 121
  moving multivolume data sets between systems 121
  REPRO command 121
  RESETCAT command 121
  RRSF VSAM file browser 245
  turning off the RACF indicator and deleting the profile 120
  using access method service commands 121
VTAM considerations for an RRSF network 149

W

WDSQUAL keyword on TARGET command 137, 160, 162
work data set for IRRUT100
  format of the records 215
worksheet, RRSF 343
workspace data sets
  defining 138, 163
  description 136
  determining how full 139
  increasing the size of 331
  maintaining 139
  naming conventions 136
  preallocating 138
  prefix for 162
  purging 173
  recovering from VSAM errors on 331
  recovering when they fill up 331
  RRSF VSAM file browser 245
  size guidelines 139
  viewing 331
WORKSPACE keyword on TARGET command 163
WTO 250

Z

z/OS UNIX System Services
  application performance 35, 75

Product Number: 5650-ZOS

SA23-2287-40