z/VM
7.4

RACF Security Server System Programmer's Guide

SC24-6312-74

Note:

Before you use this information and the product it supports, read the information in “Notices” on page 167.

This edition applies to version 7, release 4 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-29

© Copyright International Business Machines Corporation 1993, 2025.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ..... **ix**

Tables ..... **xi**

About This Document ..... **xiii**
Intended Audience xiii
Where to Find More Information xiii
Links to Other Documents and Websites xiii

How to provide feedback to IBM ..... **xv**

Summary of Changes for z/VM: RACF Security Server System Programmer's Guide ..... **xvii**
SC24-6311-74, z/VM 7.4 (September 2025) xvii
SC24-6311-74, z/VM 7.4 (June 2025) xvii
SC24-6312-74, z/VM 7.4 (September 2024) xvii
SC24-6312-73, z/VM 7.3 (December 2023) xviii
SC24-6312-73, z/VM 7.3 (September 2022) xviii

Chapter 1. Security and the RACF Database **1**
RACF and the Operating System 1
The RACF Database 2
Supporting Multiple RACF Databases 2
Backup RACF Databases 2
Shared RACF Databases 3
Location of the RACF Database 4
RACF Database on FBA DASD 4

Chapter 2. Performance Considerations **5**
Setting Up the RACF Database 5
Selecting Control Unit and Device 5
Shared RACF Database 5
Multiple RACF Databases 5
Database Housekeeping 5
Creating Backup RACF Databases 5
Resident Data Blocks 7
RVARY SWITCH Command 7
Auditing 7
Operands Requiring the AUDITOR Attribute 7
z/VM Considerations 8
MAC Filtering 9
Resetting the MAC Filter 9
Controlling z/VM Events 9
RACF Commands 10
RACF Utility Programs 10
BLKUPD 10
IRRUT200 10
Failsoft Processing 10
Installation-Written Exit Routines 11

Using Global Access Checking....11
Using the Global Minidisk Table....11
Using a Dedicated RACF Service Machine for SFS....11
The SFSAUTOACCESS Option....11
The SETROPTS Command....12
SETROPTS Command Propagation....12
Using SETROPTS RACLIST and SETROPTS GENLIST....12
Using SETROPTS INITSTATS and SETROPTS STATISTICS....15
Identification, Verification, and Authorization of User IDs....16
User Identification and Verification....17
RACHECK Processing (RACROUTE REQUEST=AUTH)....17
FRACHECK Processing (RACROUTE REQUEST=FASTAUTH)....17

Chapter 3. RACF Customization....19
Specifying RACF Database Options....19
The Database Name Table....19
The Database Range Table....22
Specifying Resource Class Options....24
The Class Descriptor Table....25
The RACF Router Table....28
Password Authentication Options....29
The RACF Encryption Algorithms (KDFAES and DES)....29
Using the Masking Algorithm....31
Using Your Own Authentication Algorithm....31
PassTicket Authentication....31
How RACF Processes the Password or PassTicket....32
Planning Considerations for Enabling KDFAES....32
Changing the ICHRSMFI Module....35
Setting the CP Disposition for Access Requests....37
Suppressing Issuance of RACF Messages....38
Defining Public Minidisks....38
Requiring Passwords for RACF Command Sessions....38
Changing User IDs for RACF Service Machines....38
Defining Multiple RACF Service Machines....39
Specifying the Value of the POSIX Constant NGROUPS_MAX....39

Chapter 4. Operating Considerations Unique to z/VM....41
Understanding RACF Interaction with CP....41
Dynamic Parse....41
Group Tree in Storage....41
Database Considerations....41
Sharing RACF Databases with Another z/VM System....43
Sharing RACF Databases in a z/VM Single System Image Cluster....43
Initializing a RACF database minidisk residing on FBA DASD....45
RACF Database on Full-pack Minidisk....45
Moving a Full-pack Minidisk to an Equally Large or Larger DASD....47
Moving User IDs and Minidisks between Systems....50
Moving User IDs....50
Moving Minidisks....51
Renaming a User....51
Modifying Applications to Use the LOGON BY Function....51
Using the GLBLDSK Macro....52
Using the SFSAUTOACCESS Option....52
Message Support....54
The Message-Routing Table....54
Enhanced Operator-Message Support....55
Using Multiple RACF Service Machines....56

Considerations for Using Multiple RACF Service Machines ... 56
Coordination of Commands ... 57
The RAC EXEC ... 57
Using RAC in the User's Virtual Machine ... 58
Modifying RAC in the User's Virtual Machine ... 59
RAC and Its EXECs in the User's Virtual Machine ... 59
Tailoring Command Output in the User's Virtual Machine ... 59
Changing Command Names and Syntax in a User's Virtual Machine ... 59
Using RAC with SFS Files and Directories ... 61
Using RAC in the RACF Service Machine ... 61
Restricting Use of RACF Commands ... 61
Adding Installation-Written Commands for the RAC Command Processor ... 62
The RACF Command Session ... 62
Adding Installation-Written Commands for the RACF Command Session ... 62
EXECs Supplied for Use on RACF for z/VM ... 63
Using ACIGROUP ... 65
ACIGROUP and Installing RACF ... 65
Adding an ACIGROUP after RACF Is Installed ... 66

Chapter 5. Utilities for the RACF Database ... 67
Format minidisk utility (RACDSF) ... 68
Allocate RACF dataset utility (RACALLOC) ... 68
RACF Security Level Display Program (RACSEC) ... 69
RACF Database-Initialization Utility Program (IRRMIN00) ... 69
Running IRRMIN00 When PARM=NEW Is Specified ... 70
Running IRRMIN00 When PARM=UPDATE Is Specified ... 70
Diagnostic Capability ... 71
Input for IRRMIN00 ... 71
Output from IRRMIN00 ... 72
RACF Cross-Reference Utility Program (IRRUT100) ... 72
Diagnostic Capability ... 73
The Work CMS File ... 73
Using IRRUT100 ... 75
RACF Database-Verification Utility Program (IRRUT200) ... 76
Copying a RACF Database ... 77
Diagnostic Capability ... 77
Processing Considerations for Databases from Other Systems ... 78
Using IRRUT200 ... 78
Utility Control Statements ... 79
Scanning the Index Blocks ... 79
BAM/Allocation Comparison ... 81
Examples of IRRUT200 usage ... 83
IRRUT200 Return Codes ... 89
The RACF Database Split/Merge/Extend Utility Program (IRRUT400) ... 89
How IRRUT400 Works ... 90
Using IRRUT400 to Extend a Database ... 90
Copying a RACF Database ... 90
Diagnostic Capability ... 90
Executing the Split/Merge/Extend Utility ... 91
Allowable Keywords ... 94
Examples of IRRUT400 usage ... 95
IRRUT400 Return Codes ... 104

Chapter 6. RACF Installation Exits ... 107
Overview ... 107
RACF Exits Report ... 107
Possible Uses of RACF Exits ... 108

Summary of Installation-Exit Callers....108
RACROUTE REQUEST=VERIFY(X) Exits....110
Preprocessing Exit (ICHRIX01)....111
Postprocessing Exit (ICHRIX02)....112
New-Password Exit....113
ICHPWX01 Processing....113
ICHPWX01 Use Case: Password Quality Control....115
Using the ICHPWX01 sample exit....115
New-Password-Phrase Exit....115
ICHPWX11 processing....116
Return Codes from the New-Password-Phrase Exit....117
Using the ICHPWX11 sample exit....118
RACROUTE REQUEST=AUTH Exits....118
Preprocessing Exit (ICHRCX01)....119
Postprocessing Exit (ICHRCX02)....120
RACROUTE REQUEST=DEFINE Exits....121
Preprocessing Exit (ICHRDX01)....121
Postprocessing Exit (ICHRDX02)....122
RACROUTE REQUEST=FASTAUTH Exits....123
Preprocessing Exit (ICHRFX01)....123
Postprocessing Exit (ICHRFX02)....125
RACROUTE REQUEST=LIST Exits....126
Pre- and Postprocessing Exit (ICHRLX01)....126
Selection Exit (ICHRLX02)....127
Data Set Naming Conventions Table (ICHNCV00)....127
Command Exits....128
ICHCNX00 Processing....129
ICHCCX00 Processing....131
Password-Encryption Exit....132
ICHDEX01 Processing....133
Modifying the ICHDEX01 Exit....134
Deleting the ICHDEX01 Exit....134
Adding the ICHDEX01 Exit....134
SMF Records Exit (ICHRSWX1)....135
RACF Report-Writer Exit....135
ICHRSMFE Processing....136
SAF Router Exits....136

Chapter 7. Recovery Procedures....139
Overview....139
Exit Routine Considerations....139
The RVARY Command....139
Failsoft Processing....141
Synchronization Considerations....142
Recovery....142
Failures on the RACF Database....142
Sample Recovery Procedures....143
Failures during RACF Command Processing....144
Commands That Do Not Modify RACF Profiles....144
Commands That Have Recovery Routines....144
Commands That Perform Single Operations....145
Commands That Perform Multiple Operations....146
Failures during RACF Manager Processing....147
Failures on the RACF Service Machine....148
When the RACF Service Machine Is Uninitialized or Unresponsive....148

Chapter 8. Storage Estimates....149

RACF Database Storage Requirements....149
Approximation of the RACF Database Size....149
System Library Storage Requirements....149
Interactive System Productivity Facility (ISPF) Storage Requirements....150
RACF Virtual Storage Requirements....150

Appendix A. Setting Up Multiple RACF Service Machines....151
Installing Multiple RACF Service Machines....151
CP Directory Considerations for Multiple RACF Service Machines....152
Initializing Multiple RACF Service Machines (RACFSVRS EXEC)....154

Appendix B. Description of the RACF Classes....155
IBM-Supplied Resource Classes that Apply to z/VM Systems....155

Appendix C. Selecting Options with ICHSECOP....157
Bypassing RACF-Initialization Processing (on z/OS)....157
Selecting the Number of Resident Data Blocks....158
Disallowing Duplicate Names for Data-Set Profiles....158

Appendix D. Using VMSES/E Support for Installation and Customization....159
Assemble a File, Modify a Build List, and Build a Library....159
Modify Full-Part ASSEMBLE and TEXT Files....160
Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List....161
Modify Full-Part Replacement TEXT Files....163
Modify Full-Part Replacement TEXT Files and Build List....164
Modify Full-Part Replacement Parts....165
Build or Link-Edit a Library....166

Notices....167
Programming Interface Information....168
Trademarks....168
Terms and Conditions for Product Documentation....168
IBM Online Privacy Statement....169

Bibliography....171
Where to Get z/VM Information....171
z/VM Base Library....171
z/VM Facilities and Features....172
Prerequisite Products....174
Related Products....174

Index....175

Figures

1. RACF and Its Relationship to the Operating System....1

2. Database Name Table Provided for z/VM Installations....20

3. ICHRDSNT Example for Three Databases....22

4. ICHRRNG Example for Three Databases....24

5. Example of CSTCONS Routing Table....54

6. RACDSF OS-Formatting Utility....68

7. RACACCOC Allocate Dataset Utility....69

8. Sample Output from IRRUT100....76

9. Sample Output of Formatted Index Produced by IRRUT200....81

10. Sample Output of Encoded BAM Map Produced by IRRUT200....83

Tables

1. Format of ICHRSMFI..... 36

2. Initial Relationships between Access Decisions Made by RACF and Final Disposition by CP.....37

3. RACDSF values..... 46

4. RACF Installation-Exits Cross-Reference Table—Part 1 of 2.....108

5. RACF Installation-Exits Cross-Reference Table—Part 2 of 2.....109

6. Fields Available during ICHPWX01 Processing.....114

7. Fields Available during ICHPWX11 Processing.....117

8. Approximate Space Requirements for the z/VM System Libraries for RACF 5.3..... 149

9. ICHSECOP Module..... 157

About This Document

This document contains information on setting up and maintaining the IBM RACF® Security Server for z/VM.

Though this document is specific to z/VM, there are references to z/OS®. These references are applicable only when sharing a RACF database with a z/OS system, which is supported only on z/VM 7.2 and earlier versions.

Intended Audience

This document is intended for the use of system programmers or installation personnel responsible for:

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

The readers of this document should be familiar with the information in:

- *z/VM: RACF Security Server General User's Guide*
- *z/VM: RACF Security Server Security Administrator's Guide*
- *RACF Program Directory for z/VM*

z/VM: RACF Security Server Auditor's Guide may also be useful.

Where to Find More Information

For information about related publications, refer to the "Bibliography" on page 171.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

© Copyright IBM Corp. 1993, 2025

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

© Copyright IBM Corp. 1993, 2025

Summary of Changes for z/VM: RACF Security Server System Programmer's Guide

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

SC24-6311-74, z/VM 7.4 (September 2025)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.4.

RACF utility enhancements

With the PTF for APAR VM66873, z/VM 7.4 provides the ability to use RACF utilities RACUT200 and RACUT400 non-interactively (by passing them input parameters from the command line) rather than interactively (by responding to system prompts, as was previously required). The amount of input that users must enter is reduced by setting default values for certain parameters and by allowing abbreviations for certain parameters.

The following topics are updated:

- “Copying a non-shared RACF database” on page 84
- “Copying a shared RACF database” on page 84
- “Making a Point-In-Time Backup Copy of the Primary RACF Database” on page 85
- “Verifying a database with RACF active” on page 88
- “Verifying a database with RACF inactive” on page 89
- “Executing the Split/Merge/Extend Utility” on page 91
- “Allowable Keywords” on page 94
- “Copying a RACF database to a larger volume without shutting down the RACFVM server” on page 95

Miscellaneous updates

The following topics are updated:

- “Database Considerations” on page 41
- “Sharing RACF Databases in a z/VM Single System Image Cluster” on page 43
- “Initializing a RACF database minidisk residing on FBA DASD” on page 45
- “Format minidisk utility (RACDSF) ” on page 68
- “Allocate RACF dataset utility (RACALLOC) ” on page 68

SC24-6311-74, z/VM 7.4 (June 2025)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.4.

SC24-6312-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

© Copyright IBM Corp. 1993, 2025

SC24-6312-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6312-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

[7.3] RACF support for z/VM 7.3

Select RACF utilities for database installation, maintenance, and operations along with select RACF reports are now allowed to run if the RACF service machine's 490 minidisk was IPLed; IPLing CMS is no longer required. In addition, The **RACUT100**, **RACUT200**, and **RACFCONV** utilities require the IPL of the 490 minidisk to support reserve and release of the RACF database.

The following topics are updated:

- “Dynamic Parse” on page 41
- “Example of the RACUT400 EXEC (Splitting Four Databases)” on page 102
- “Executing the Split/Merge/Extend Utility” on page 91
- “Making a Point-In-Time Backup Copy of the Primary RACF Database” on page 85
- Splitting One Database into Four Databases: “Step Five” on page 100

RACF database sharing with z/OS is not supported on z/VM 7.3 and later versions. After the RACF database template is upgraded for z/VM 7.3 by using the RACFCONV utility, database sharing is not supported. Database sharing is supported on z/VM 7.2 and earlier versions and is documented in z/VM 7.2 and earlier versions publications.

Chapter 1. Security and the RACF Database

RACF Security Server for z/VM is a product that works together with the existing system features of z/VM.

RACF helps meet the 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.

As system programmer, it is your responsibility to implement the installation's security goals. You need to:

- Provide technical input on the feasibility of the implementation plan
- Install RACF on the system
- Provide technical support for RACF
- Maintain the RACF database
- Oversee the programming aspects of system protection

In addition, you may need to write, install, and test RACF installation exit routines.

RACF and the Operating System

To visualize how RACF works, picture RACF as a layer in the operating system that verifies users' identities and grants user requests to access resources.

Assume, for example, that you have been identified and verified to the RACF-protected system and now want to modify an existing RACF-protected resource. After you enter a command to the system to access the resource, a system resource manager (such as data management) processes the request. The resource manager, based on what RACF indicates, either grants or denies the request.

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

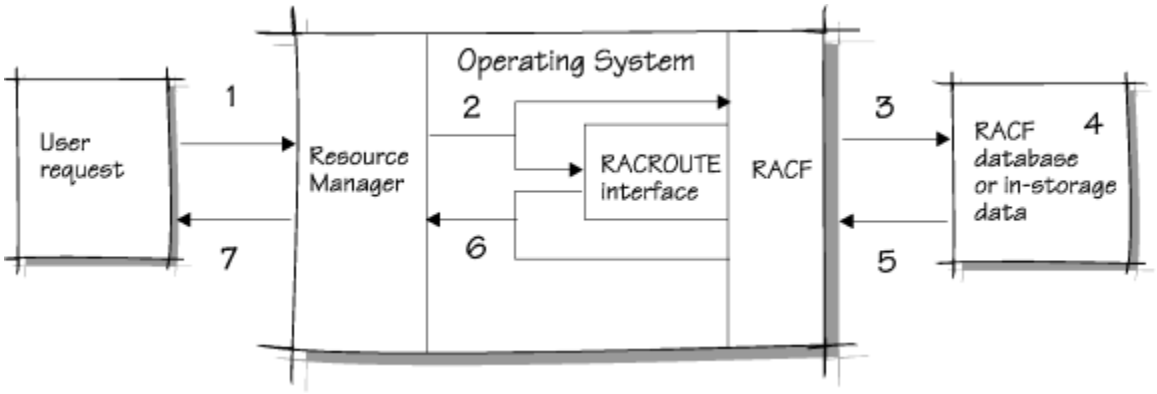


Figure 1. RACF and Its Relationship to the Operating System

1. A user requests access to a resource using a resource manager (for example, SFS).
2. The resource manager issues a RACF request to see if the user can access the resource.
3. RACF refers to the RACF database or in-storage data and...
4. ...checks the appropriate resource profile.

© Copyright IBM Corp. 1993, 2025

5. Based on the information in the profile...
6. RACF passes the status of the request to the resource manager.
7. The resource manager grants (or denies) the request.

The RACF Database

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.

RACF 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/VM: RACF Security Server Diagnosis Guide*.

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

Supporting Multiple RACF Databases

Multiple RACF database support provides an alternative to maintaining all your RACF profiles on one database. By dividing your database, you reduce device contention and improve performance. If one device experiences an I/O failure, the others may be unaffected.

RACF allows you to provide a backup database to which you can switch without a re-IPL should your primary RACF database fail.

You can also share your RACF database with another operating system.

Note: RACF for z/VM does not allow multiple RACF databases on FBA devices.

Also, see the program directories and Chapter 8, “Storage Estimates,” on page 149.

Backup RACF Databases

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 your primary databases, or some of them, depending on the needs of your installation. Use the RACF database name table (ICHRDSNT) to control the amount of updating to the backup database. For information on ICHRDSNT, see “The Database Name Table” on page 19.

RACF allocates the backup databases at the same time it allocates the primary databases; therefore, the backup databases must be online during RACF initialization. In case of a failure on a primary RACF database, RACF enables you to switch to your backup RACF databases without having to re-IPL. The primary and backup should reside on different real devices and on different paths. For a discussion of how to handle database failures, see Chapter 7, “Recovery Procedures,” on page 139.

For additional information on backing up your database, see “Creating Backup RACF Databases” on page 5.

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 or IRRUT400 utilities. See “Copying a non-shared RACF database” on page 84 under “Examples of IRRUT200 usage” on page 83 and

"Copying a RACF database to a larger volume without shutting down the RACFVM server" on page 95 under "Examples of IRRUT400 usage" on page 95. Both IRRUT200 and URRUT400 can produce only disk output; for tape, you should provide an additional copy step. RACF databases and all copies should be RACF-protected at all times.

Shared RACF Databases

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

- z/VM running native
- z/VM running as a guest machine of z/VM
- z/VM running on a system in an SSI cluster

When the RACF database is to be shared, the device on which the database resides must be system-generated as shared, or damage to the database is likely. Both primary and backup databases must be shared.

For information on how to system-generate a device as shared, see *z/VM: I/O Configuration*.

The RACF database must match the latest level of code. The IRRMIN00 utility (invoked by the RACFCONV EXEC on z/VM) is used to update the database when a new release or service level is available.

- The database is downwardly compatible. For example, if RACF 1.10 and RACF 5.3 are sharing a database, that database must be at the 5.3 level, but your 1.10 system can successfully use it.

Note: RACF database sharing with z/OS is not supported on z/VM 7.3 and later versions. After the RACF database template is upgraded for z/VM 7.3 by using the RACFCONV utility, database sharing is not supported. Database sharing is supported on z/VM 7.2 and earlier versions and is documented in z/VM 7.2 and earlier versions publications.

General Considerations

If you choose to use the data encryption standard (DES) algorithm in encrypting RACF passwords, you must use it on all the systems that share the RACF database. Your encryption method must be the same on all systems sharing the database.

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

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

All sharing systems must have compatible class descriptor tables (ICHRRCDE); they do not need to be identical, but there can be no conflicts in them.

z/VM Considerations

If you wish to share your RACF database with a native z/VM system, you cannot place your RACF database on a z/VM minidisk; you must place it either on a dedicated device or on a full-pack minidisk.

The requirements for sharing volumes depend on the configuration of the system. For example, sharing the database with a second-level guest requires the use of MWV on the MDISK statement for the database.

For more information, see "Database Considerations" on page 41.

All systems that share databases must refer to those databases by the same name.

- If all systems are z/VM systems and default database names have been used, the names are the same.
- If different database names have been used, changes must be made in change the ICHRDSNT database name table to make the database names identical. Database names in ICHRDSNT must have the same names as those of the existing databases.

The RACF database cannot be shared if it resides on an FBA device. If you are sharing a RACF database with a z/VM system, you cannot have an alternative path to the database online.

Attention

You must add the V suffix when you define each database minidisk in the user directory. The V-suffix configuration is required for concurrent RACF database maintenance, even if the database is not shared with another z/VM system. If the RACF databases are not defined with the V suffix on the minidisk link mode (e.g. "MWV"), then RACF reports the following error and immediately logs off to prevent damage to the RACF databases:

```
CSTERP001E Minidisk ... was defined without the V suffix in the CP directory (e.g. MWV).
```

See “Sharing RACF Databases with Another z/VM System” on page 43.

Location of the RACF Database

On z/VM

RACF databases can reside on several types of DASD devices. Consult *RACF Program Directory* for a list of the supported DASD devices. If RACF is heavily used and you elect to use a single RACF database, plan to put the database on a device accessed by the channel and control unit least likely to affect system performance.

The RACF database is on a minidisk, unless it is to be shared with an z/OS system. (See “Shared RACF Databases” on page 3.) After the RACF service machine initializes, you cannot move the RACF database to another location on the same pack without causing I/O errors. If you move the database, you must re-IPL each RACF service machine to access the RACF database again.

RACF Database on FBA DASD

The minidisks for the RACF database (defined at virtual addresses 200 and 300) can reside on FBA devices. The FBA devices must be supported by z/VM and include devices 3370, 9332, 9335, or 9336. (For the most current information on which FBA DASD devices are supported by z/VM refer to *z/VM: CP Planning and Administration*.)

Restrictions

- The RACF database cannot be shared on FBA device types.
- Multiple RACF service machines cannot be used when the RACF database resides on FBA device types.
- The primary RACF database must have SYSRACF as the ddname in its FILEDEF statement. The backup RACF database must have RACFBKUP as the ddname in its FILEDEF statement.
- The number of RACF databases is limited to a primary and a backup. The use of the backup database is optional.

Chapter 2. Performance Considerations

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

Setting Up the RACF Database

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

Selecting Control Unit and Device

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

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 may 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.

Multiple RACF Databases

If you divide the RACF database into multiple databases, you can reduce the importance of the preceding factors, because:

- Each RACF database may receive fewer requests
- Each RACF database may be smaller, in which case each request requires fewer I/O requests and fewer movements of the activator arm on the device
- Each RACF database 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 7.

Database Housekeeping

Organizing 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 for a database:

- IRRUT200 creates an exact block-by-block copy of the RACF database. This exact copy can help performance when you are maintaining statistics on your backup database.

  Note: IRRUT200 does not serialize on the RACF database and should not be used when there is update activity on the database being copied. The following precautions should be observed in order to maintain data integrity of the target database:

© Copyright IBM Corp. 1993, 2025

  - IRRUT200 copy should not be used in a shared database environment.
  - IRRUT200 copy should be run from the RACFVM server virtual machine with RACF deactivated.

  Copying a database that is active and being updated while the copy is in progress may compromise the data integrity of the target database. IRRUT200 can be used only if you are creating a backup database that is the same size and on the same device type as the input database. For more information, see "RACF Database-Verification Utility Program (IRRUT200)" on page 76.
- IRRUT400 creates a copy of your database and can be used to change its size.

  IRRUT400 also reorganizes the contents of the output RACF database. Use this utility if you are copying between different device types. You can also use IRRUT400 to extend the RACF database before it becomes full. For important information, see "The RACF Database Split/Merge/Extend Utility Program (IRRUT400)" on page 89.

Options for Updating Backup Databases

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

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

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

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

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

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

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

   We recommend that you use this option in your database 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 Database Name Table" on page 19.

Resident Data Blocks

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

For each primary RACF database, an installation can assign from 0 to 255 resident data blocks. The default value is 100 resident data blocks. For best performance, specify as large a number of buffers as you can afford, preferably 255. The storage is in RACFVM's virtual storage.

You can have resident data blocks when the RACF database resides on a shared device. RACF updates the resident data blocks to ensure that all processors use the latest level of the blocks. When resident data blocks are used for a shared RACF database, some resource statistics may not be updated. For more information, see “Selecting the Number of Resident Data Blocks” on page 21.

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).

The database buffers are switched when the databases are both on shared devices or both on non-shared devices.

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 impacts 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 | NOAPPLAUDIT
- AUDIT | NOAUDIT
- CMDVIOL | NOCMDVIOL
- LOGOPTIONS
- OPERAUDIT | NOOPERAUDIT
- SAUDIT | NOSAUDIT
- SECLABELAUDIT | NOSECLABELAUDIT
- SECLEVELAUDIT | 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.

Shared file system (SFS) invokes RACROUTE REQUEST=DEFINE on delete, rename, and relocate functions for SFS files and directories and produces an SMF record for each of these functions.

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

z/VM Considerations

Several auditing performance issues are unique to the z/VM environment.

VMXEVENT Auditing

The VMXEVENT class enables you to audit all z/VM events, such as DIAG0D4, TRANSFER, LOGON, and LINK. Be careful when you audit commonly used commands, such as MESSAGE or DIAG008. Auditing events can affect system performance. If you choose to audit the TRANSFER event, an audit record is

written for **each** file transferred as a result of the TRANSFER and CHANGE TO commands. This auditing occurs even for privileged users, such as on the TRANSFER SYSTEM ALL command.

Auditing for Individual Users

Use of individual z/VM event profiles enables an installation to monitor an individual user's activities. Individual auditing can reduce performance overhead because auditing is done not on a system-wide basis, but just for a given user.

Number of Audit Buffers

The number of audit records RACF buffers before they are written to the SMF file can affect performance. This value is specified in the SMF CONTROL FILE; the default value is CLOSE 001, but it can be as high as 999. If the system goes down, the records in the buffer are lost. An installation must weigh its performance needs against its auditing requirements.

MAC Filtering

The mandatory access control (MAC) filtering capability of RACF allows it to make some access decisions directly within its CP modules. Because a large amount of security label comparison requests are generated by z/VM when the SECLABEL class is active, avoiding the IUCV and service machine overhead required to process these events will reduce performance degradation that may otherwise occur.

To avoid the increased system overhead that occurs when the MAC filter is not used, the following items should be considered:

- The MAC filtering function is bypassed if any of the following are true:
  - `SETROPTS LOGOPTIONS(SUCCESSES(VMMAC))` is in effect
  - `SETROPTS LOGOPTIONS(FAILURES(VMMAC))` is in effect
  - `SETROPTS LOGOPTIONS(ALWAYS(VMMAC))` is in effect
  - `SETROPTS MLQUIET` is in effect
- The effectiveness of the MAC filtering function is reduced if any of the following are true:
  - `SETROPTS MLS(WARN)` is in effect
  - `SETROPTS MLACTIVE(WARN)` is in effect
  - Some security labels are being audited with the `SETROPTS SECLABELAUDIT` option

Resetting the MAC Filter

RACF resets the filter whenever the SETROPTS command is issued in order to ensure that the MAC filter does not become back-level in regards to changing SECLABEL definitions. This causes a period during which RACF performance may be degraded in relation to performance when the filter is fully populated.

Likewise, the filter will be reset whenever a RACF service machine is initialized.

Controlling z/VM Events

Access checking enabled by VMXEVENT settings (control is ON) may contribute to performance overhead. The performance overhead will not be appreciable, however, unless the z/VM event being controlled is performed often.

You may want to specifically turn off control for some of the commands that are issued often and are controlled by default. Doing this eliminates an extra call to the RACF service machine. Events you may want to consider for turning off control are: TAG, TRANSFER.G, TRANSFER.D and MDISK.

For information on how to turn control off by defining VMXEVENT profiles, refer to *z/VM: RACF Security Server Security Administrator's Guide*.

RACF Commands

Any RACF command that requires reading or processing a large number of profiles (for example, SEARCH, LISTDSD with the ID or PREFIX operands, LISTGRP *, LISTUSER *, and RLIST class-name *) can cause contention for the RACF database. These commands repeatedly issue reserves each time a profile is processed.

Depending on the amount of contention, the processing of other RACF commands can be slowed down, and systems sharing the RACF database can appear to be locked out. This contention may be reduced if the resident data-block option is in effect, and if the data can be located in one of the blocks.

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. The RACFDBU EXEC invokes the IRRDBU00 utility. For information on IRRDBU00 and RACFDBU, see *z/VM: RACF Security Server Security Administrator's Guide*.

RACF Utility Programs

When one of the RACF utility programs is processing a single RACF database, that RACF database can 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/VM: RACF Security Server Security Administrator's Guide* and *z/VM: RACF Security Server Macros and Interfaces*.

BLKUPD

The use of the BLKUPD utility command can cause long reserves against the RACF database because the reserves are maintained 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 obtain a working copy of the RACF database defined by the SYSUT1 DD statement on z/OS and the RACONFIG EXEC on z/VM. If you use the copy function, IRRUT200 uses the copied database, and the RACF database is available during subsequent processing of IRRUT200.

For more information on the utilities, see Chapter 5, “Utilities for the RACF Database,” on page 67.

Failsoft Processing

Failsoft processing occurs when there are no primary RACF databases available (RACF is installed but inactive). It degrades system performance and system security.

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

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

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 RACHECK and RACDEF installation exits to implement failsoft processing in some other way.

See also Chapter 7, “Recovery Procedures,” on page 139.

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 6, “RACF Installation Exits,” on page 107.

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/VM: RACF Security Server Security Administrator's Guide*.

Using the Global Minidisk Table

You can use the global minidisk table to improve performance by not calling the RACF service machine for public minidisks. The global minidisk table should be used for minidisks that are accessed frequently in R or RR mode by large numbers of users. The minidisks should not contain any installation-sensitive data.

The global minidisk table is maintained in storage in the RACF module HCPRWA. If an entry in the global minidisk table allows the access to a minidisk, the RACF service machine is not called. This eliminates the overhead associated with IUCV activity and can result in performance improvements.

See “Using the GLBLDSK Macro” on page 52 for more information on creating a global minidisk table. For a complete description of the macro, see *z/VM: RACF Security Server Macros and Interfaces*.

Using a Dedicated RACF Service Machine for SFS

When RACF is specified as the external security manager for shared file system (SFS), the number of calls to the RACF service machine increases dramatically. For this reason, IBM recommends that you dedicate at least one RACF service machine to process RACROUTE requests from the SFS file pool server. If you have more than one SFS file pool server, you can:

- Send all RACROUTE requests to one RACF service machine
- Assign groups of them to different RACF service machines

For more information, see “Dedicating RACF Service Machines for RACROUTE Request Processing” on page 57.

The SFSAUTOACCESS Option

To reduce the number of RACROUTE calls, the RACF SERVMACH file uses the SFSAUTOACCESS option. It allows the RACROUTE REQUEST=AUTH interface running in the SFS file pool server to grant access to a file or directory automatically to users who are accessing their own SFS files or directories. For more information, see “Using the SFSAUTOACCESS Option” on page 52.

The SETROPTS Command

Certain operands of the SETROPTS command directly affect system performance:

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

SETROPTS Command Propagation

If you issue the SETROPTS command with any operand that changes the RACF database or issue the SETROPTS REFRESH command, the command is automatically propagated to all RACF servers that run on the same z/VM system, and to other systems in the same SSI cluster as the issuing system. Only the SETROPTS LIST command is not propagated.

On a system outside an SSI cluster, the action is not propagated to other systems that share the RACF database. You must issue the SETROPTS command separately for each system or restart the RACF servers on the other system or IPL the other system.

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 from the RACF database into storage. Issuing a SETROPTS RACLIST minimizes I/O.

Restriction:

You cannot use SETROPTS RACLIST processing with the following classes:

DIRECTRY
FILE
SFSCMD
VMMDISK
VMPOSIX
VMRDR

The GENLIST operand on the SETROPTS command improves performance by copying generic profiles from the RACF database into storage. If you are using generic profiles, it is desirable to share these profiles so that just one copy of the profiles is kept in each service machine. The copy is then available to all users who require access to a resource the profile protects.

We recommend that you issue a SETROPTS GENLIST command for the VMMDISK, VMBATCH, VMNODE, and VMRDR classes.

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

- Whether you can afford the storage utilization.
- 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.

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

RACLIST Processing

The RACLIST operand on the SETROPTS command copies generic and discrete profiles from the RACF database into the RACF service machine's virtual storage.

RACF uses these profile copies to check the authorization of any user who wants to access a resource protected by them.

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 a small number of 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; the global access table is ignored for any RACLIST class.

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

```
SETROPTS RACLIST(class name...) CLASSACT(class name...)
```

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

APPCTP	CSFSERV	OPERCMDS	PSFMPL	SECLABEL
APPCSERV	DEVICES	PROPCNTL	RACFVARS	VTAMAPPL
CSFKEYS	NODES	PTKTDATA		

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

ACCTNUM	APPCSI	APPCPORT	APPL	CONSOLE
DASDVOL	DLFCLASS	DSNR	FACILITY	FIELD
FCICSFCT	INFOMAN	JESINPUT	JESJOBS	JESSPOOL
LFSCLASS	MGMTCLAS	MQCMDS	MQCONN	PERFGRP
SDSF	SMESSAGE	STORCLAS	SURROGAT	TERMINAL
TSOPROC	TSOAUTH	VMBATCH	VMCMD	VMLAN
VMNODE	VMSEGMT	WRITER		

See “SETROPTS Command Propagation” on page 12 for information on the SETROPTS commands that are automatically propagated in certain system environments.

Field-Level Access Checking

If you choose to use field-level access checking to control access to profile fields, you can improve system performance if you issue a SETROPTS RACLIST for the FIELD class. When you use RACLIST processing for the FIELD class, RACF brings copies of the profiles that protect the fields into common storage.

Because the number and size of the profiles in the FIELD class is normally small, there should not be a large impact on the use of common storage.

For more information on field-level access checking, see *z/VM: RACF Security Server Command Language Reference* and the *z/VM: RACF Security Server Security Administrator's Guide*.

RACROUTE Considerations When Using SETROPTS RACLIST

If you use RACROUTE REQUEST=AUTH for authorization checking, the profiles that were brought into storage with the SETROPTS RACLIST command are accessible.

If you use RACROUTE REQUEST=FASTAUTH, you cannot use profiles that were brought into storage with the SETROPTS RACLIST command. When you use RACROUTE REQUEST=FASTAUTH, you must get the profiles you need by using RACROUTE REQUEST=LIST.

Your installation should not issue a SETROPTS RACLIST for any class for which a RACROUTE REQUEST=LIST macro was entered. Doing a SETROPTS RACLIST in this case would only waste storage.

Refreshing SETROPTS RACLIST Processing

REFRESH RACLIST causes all the in-storage discrete and generic profiles for the resource to be replaced with new copies from the RACF database.

If SETROPTS RACLIST has been issued for a general resource class and you change the general resource profile in the class, you may want to use REFRESH RACLIST to refresh the profile.

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

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

Shared System Considerations

In a non-SSI cluster environment, the refresh operation for SETROPTS RACLIST processing applies only to the system on which you issue the SETROPTS command. If your installation has two or more non-cluster systems sharing a RACF database, you must issue the SETROPTS command on all systems to have the refresh done on all systems. 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. See “SETROPTS Command Propagation” on page 12 for information on the SETROPTS commands that are automatically propagated in certain system environments.

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 the RACFVM service machine. Using GENLIST saves real storage because there is only one copy in the service machine.

RACF uses these profile copies to check the authorization of any user who wants to access a resource the profiles protect.

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

```
SETROPTS GENLIST(class name...) CLASSACT(class name...)
```

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 RACF servers and 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 IBM-supplied classes can be shared by using SETROPTS GENLIST:

APPL	FACILITY	INFOMAN	TERMINAL	VMMDISK
DASDVOL	FIELD	JESJOBS	VMBATCH	VMNODE
DSNR	SDSF	VMCMD	VMLAN	VMRDR
VMSEGMT				

Considerations Unique to z/VM

You should issue a SETROPTS GENLIST for those classes that contain generic profiles. If you have classes that contain a small number of profiles (both generic and discrete), it is better to issue a SETROPTS RACLIST for the class rather than a SETROPTS GENLIST.

Refreshing In-Storage Generic Profiles

You may 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

In a non-SSI cluster environment, the refresh operation for SETROPTS processing applies only to the system (z/VM or z/OS) on which you issue the SETROPTS command. If your installation has two or more non-cluster systems sharing a RACF database, you must issue the SETROPTS command on all systems to have the refresh done on all systems. 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. See “SETROPTS Command Propagation” on page 12 for information on the SETROPTS commands that are automatically propagated in certain system environments.

Using SETROPTS INITSTATS and SETROPTS STATISTICS

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

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

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

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

  For details, see the SETROPTS command in *z/VM: RACF Security Server Command Language Reference*.

Note: If you are sharing a database and are using in-storage data blocks, statistical information may 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.

Recommendations on Using INITSTATS

Although INITSTATS affects performance because of I/O to the database, it is recommended that INITSTATS stay on. You can then use the SETROPTS operands INACTIVE and PASSWORD. For additional information, see *z/VM: RACF Security Server 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. 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/VM: RACF Security Server 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 may 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 may 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 may decrease sharply.

See the SETROPTS command in *z/VM: RACF Security Server 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=AUTH

User Identification and Verification

RACINIT Processing (RACROUTE REQUEST=VERIFY or VERIFYX)

The RACINIT function does identification and verification of users and determines whether work is allowed to enter the system. Some of the events that can cause RACINIT processing to occur are:

- Logons
- Validating passwords or password phrases using diagnose A0, diagnose 88, or APPC
- Sending data sets to the printer (if WRITER class is active)
- authenticating to various TCP/IP applications such as FTP and TELNET
- Entering a RACF command session

Some of the checks done by RACINIT processing are:

- Surrogate checking
- Terminal-authorization and port-of-entry checking
- SECLABEL checking

RACHECK Processing (RACROUTE REQUEST=AUTH)

Whenever a user attempts to access a resource, the system calls RACF to perform authorization checking. During normal RACHECK 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 RACHECK 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. Global access checking is ignored for RACLIST classes.

On z/VM

For systems with many shared resources, the best way to improve performance is to place the frequently referenced resources into the global access table. For example, the network virtual machine's reader is accessed many times a day. System performance can be improved considerably if you bypass normal RACHECK processing for this resource, and place the network virtual machine's unit record devices in the global access table. You should also place frequently accessed minidisks such as the S-disk and the systems disk, 19E, in the global access table or the global disk table. See *z/VM: RACF Security Server Macros and Interfaces* for more information on the global disk table.

FRACHECK Processing (RACROUTE REQUEST=FASTAUTH)

RACROUTE REQUEST=FASTAUTH uses the resident profiles constructed by the RACROUTE REQUEST=LIST macro to perform authorization checking. RACROUTE REQUEST=FASTAUTH performs no logging, 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. RACROUTE REQUEST=FASTAUTH is particularly useful for applications that have stringent performance requirements. However, RACROUTE REQUEST=FASTAUTH processing does complicate your application coding: You must use RACROUTE REQUEST=LIST to load profiles into storage and to delete them when you are done. In addition, if your application is long-running, you may need to supply a "refresh" mechanism in case the security administrator has changed your profile.

Chapter 3. RACF Customization

Specifying RACF Database Options

You can specify options for the RACF database by using:

- The database name table, which describes the RACF databases to RACF and allows you to select the number of resident data blocks
- The database range table, which determines the RACF database to be accessed for a particular profile.

The Database Name Table

The database name table (ICHRDSNT) is a customer-provided load module that describes the RACF databases to RACF. This table contains entries describing each RACF database and its backup database.

A database's position in this table corresponds to the database number in the range table. If a database is named in the database name table, it *must* be referenced in the range table. If the name table does not match (that is, has more entries than) the range table, RACF does not become active during the IPL.

On z/OS, RACF can have as many as 90 primary databases and 90 associated backup databases. On z/VM, RACF can have four primary databases and four associated backup databases.

Note: RACF for z/VM does not allow multiple RACF databases on FBA devices.

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

A 44-byte database name (primary)

The database name identifies the primary database. The primary RACF database is considered to be the “master” RACF database. If your installation has specified this field with an asterisk (*), RACF prompts the operator, during RACF initialization, to supply the database name.

A 44-byte database name (backup)

The second database name identifies an associated backup database. If your installation has specified this field with an asterisk (*), RACF prompts the operator again. A blank database name field indicates the absence of either a primary or backup database, or both, for this IPL.

A 1-byte resident data-block count field

The resident data block count field specifies the maximum number of index, block-availability-mask (BAM), and profile blocks to be kept resident for the primary database while it is active. See “Selecting the Number of Resident Data Blocks” on page 21.

A 1-byte flag field

The format of the flag field is as follows:

Bit Setting
Meaning

© Copyright IBM Corp. 1993, 2025

00.. ....
No updates are to be duplicated on the backup database. This is the default setting if you do not provide ICHRDSNT.

10.. ....
All updates, but no statistics, are to be duplicated on the backup database. This is the recommended setting if you provide ICHRDSNT.

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

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

SETROPTS INITSTATS allows 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, and therefore the statistics for the first logon of the day will not be recorded in the backup database.

11.. ....
All updates, including statistics, are to be duplicated on the backup database.

....1
Controls the resident data-block option for the primary database. This option is always used and the bit setting is ignored.

z/VM Considerations

A database name table (ICHRDSNT ASSEMBLE) is provided on the product tape (see Figure 2 on page 20) and resides in the RACFLINK LOADLIB on RACFVM's 305 disk. This table assumes one primary RACF database named RACF.DATASET and a backup RACF database named RACF.BACKUP. There are 100 (X'64') resident blocks indicated in the table. The flag field is set to X'81', meaning that all updates other than statistics updates are to be duplicated in the backup database. The resident data blocks include both index and non-index blocks.

```
ICHRDSNT CSECT
*********************************************************************
*        ICHRDSNT - RACF DATA SET NAME TABLE
*********************************************************************
         SPACE 2
*********************************************************************
*      NUMBER OF NAME PAIR ENTRIES
*********************************************************************
         SPACE 2
         DC    X'01'
         SPACE 2
*********************************************************************
*        DATA SET NAMES
*        FORMAT:
*              CL44'PRIMARY DSNAME',CL44'SECONDARY DSNAME',X'N1',X'N2'
*
*                    N1=NUMBER OF RESIDENT BLOCKS
*                    N2=FLAG FIELD
*                         BIT 0 = UPDATES ARE TO BE DUPLICATED
*                         BIT 1 = STATISTICS ARE TO BE DUPLICATED
*                         BIT 7 = USE RESIDENT DATA BLOCK OPTION
*
*
*********************************************************************
         SPACE   3
*********************************************************************
NAME1    DC    CL44'RACF.DATASET',CL44'RACF.BACKUP',X'64',X'81'
*********************************************************************
         END   ICHRDSNT
```

Figure 2. Database Name Table Provided for z/VM Installations

Each database name in the database name table corresponds to a virtual address in the RACONFIG EXEC. The virtual address for the primary is assigned to the variable `racf_primary_disk`. The virtual address for the backup is assigned to the variable `racf_backup_disk`.

Selecting the Number of Resident Data Blocks

In the database name table (ICHRDSNT), you can specify the number of resident data blocks for each primary RACF database. This keeps any type of data block (profile and BAM blocks as well as index blocks) resident. An installation can specify from 0 to 255 resident data blocks; the default value is 100. 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. This function is also dynamic because, at any time, only the most frequently used blocks are kept in storage.

On the RACF database, each data block uses 4128 (4KB + 32) bytes of storage. The storage is in RACFVM's virtual storage.

At RACF initialization, ICHSEC00 obtains the storage for the number of buffers specified in the database name table. The RACF manager then keeps track of when each buffer was used last. The RACF manager does different processing for shared and non-shared databases.

Shared RACF Database

- The change count in the inventory control block (ICB), corresponding to the block type (profile or index), is updated whenever a block is updated.
- 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.
- 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 may 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.

Non-shared RACF Database

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

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

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

Database Name Table Example for a Split Database

Assume that your database has been split into three parts and that your installation arranges its database profiles in the following manner:

- 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 database for each primary RACF database. However, the backup of the databases is different:

- For RACF.RACFDS1, all updates to the primary database, except statistics, will be duplicated in the backup database.
- For RACF.RACFDS2 and RACF.RACFDS3, all updates to the primary database, including statistics, will be duplicated in the backup database.

This database name table correctly follows this criterion:

AL1(3)
Number of primary RACF databases

CL44'RACF.RACFDS1'
Name of first RACF database (test data sets)

CL44'RACF.BACKUP1'
Name of first RACF database backup

AL1(20)
Number of resident blocks

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

CL44'RACF.RACFDS2'
Name of second RACF database (production data sets)

CL44'RACF.BACKUP2'
Name of second RACF database backup

AL1(10)
Number of resident blocks

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

CL44'RACF.RACFDS3'
Name of third RACF database (system data sets)

CL44'RACF.BACKUP3'
Name of third RACF database backup

AL1(255)
Number of resident blocks

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

Figure 3. ICHRDSNT Example for Three Databases

Modifying the Database Name Table

To modify the database name table, you need to perform local modifications to ICHRDSNT ASSEMBLE and ICHRDSNT TEXT. Follow the instructions in "Modify Full-Part ASSEMBLE and TEXT Files" on page 160, using these substitution values.

- For *fn* use **ICHRDSNT**
- For *nnnn* use **0002**

The Database Range Table

The range table (ICHRRNG) is a load module. This table determines on which RACF database to place each profile. The table resides in RACFLPA LOADLIB on RACFVM's 305 disk.

Using these values, RACF provides a default range table:

F'1'
Number of range values

XL44'000—000'
Range start value

AL1(1)
Database number

This table assumes a single RACF database containing all profiles. If you wish to split your database, you must replace the RACF load module with your own. This is done by creating a source file, assembling the file, and link-editing it.

Table Format

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

XL44'000—000'
Range start value

AL1(n)
where n is the database 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 **database number** indicates the database's relative position in the database name table (ICHRDSNT).

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

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

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

Internal Profile Naming

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. On z/VM, a VMMDISK named MY.191 becomes VMMDISK -MY.191.

RACF modifies generic profile names internally, as follows:

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

Database Range Table Example

An installation wants all profile names beginning with “GRPA” through “GRPF” and “TEST1” through “TEST8” to reside on database 2, all profile names beginning with “SYS1” to reside on database 3, and all the remaining data to reside on database 1. This range table meets the criteria:

F'7'
Number of range values

XL44'000—000'
Range start

AL1(1)
database number

XL44'C7D9D7C1000—000'*
Range start GRPA

AL1(2)
database number

XL44'C7D9D7C7000—000'
Range start GRPG

AL1(1)
database number

XL44'E2E8E2F1000—000'
Range start SYS1

AL1(3)
database number

XL44'E2E8E2F2000—000'
Range start SYS2

AL1(1)
database number

XL44'E3C5E2E3F1000—000'
Range start TEST1

AL1(2)
database number

XL44'E3C5E2E3F9000—000'
Range start TEST9

AL1(1)
database number

Figure 4. ICHRRNG Example for Three Databases

* You must pad profile names to 44 characters with binary zeros.

Modifying the Database Range Table

To modify the database range table, you need to perform local modifications to ICHRRNG TEXT. Follow the instructions in "Modify Full-Part Replacement TEXT Files" on page 163, using the following substitution values. Note that you are building the RACFOBJ TXTLIB and RACFLPA LOADLIB libraries.

- For *fn* use **ICHRRNG**
- For *nnnn* use **0002**

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. For each general resource class, there is a unique entry in the table.

The Class Descriptor Table

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

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

- ICHRRCDX is the name for the IBM-supplied class entries.
- ICHRRCDE is the name for the installation-defined class entries.

Each IBM-supplied class 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 IBM-supplied classes, see Appendix B, "Description of the RACF Classes," on page 155.

RACF processing references the 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 both systems, it must have the same attributes. For example, the POSIT numbers must be the same. If you have different releases of RACF (5.3 and 1.10), you can share a database without adding the new 5.3 classes to the class descriptor table on the 1.10 system. RACF utilities must not be used from the down-level system.

An installation can add, modify, or delete installation-defined class entries. The ICHERCDE macro cross-checks class-descriptor-table entries 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.

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

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

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

Installation-Defined Classes and the RACF Router Table

If a class entry is added to the installation's class descriptor table, a corresponding entry should be added to the RACF-router table (using the ICHRFRTB macro). A discussion of ICHRFRTB follows in this chapter.

Adding Installation-Defined Classes

The ICHERCDE macro generates class entries for the RACF class descriptor table. Each installation-defined class-descriptor-table entry becomes a CSECT in load module ICHRRCDE. The module resides in RACFLINK LOADLIB on RACFVM's 305 production build disk.

To add a class entry, use the ICHERCDE macro to create a new class entry. The steps that follow describe in detail how to:

- Create a new text file with your class entry in it
- Update the RACFLINK LOADLIB to include your class entry

1. Log on to the MAINT740 user ID.
2. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

For *compname*, use:

RACF
For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

3. Create assembler program ICHRRCDE ASSEMBLE on the local modification LOCALSAM 2C2 disk.

   The following is a sample of code that adds one installation-defined class in ICHRRCDE.

```
        ICHRRCDE CSECT
        $USRCLS  ICHERCDE CLASS=$USRCLS,
                       ID=130,
                       MAXLNTH=246,
                       FIRST=ANY,
                       OTHER=ANY,
                       POSIT=19,
                       OPER=YES,
                       RACLIST=ALLOWED,
                       GENLIST=ALLOWED,
                       DFTUACC=NONE
                 ICHERCDE
                 END
```

   Notice that the last entry of ICHERCDE in the example does not have any operands.

   Note: Where there are commas, there must also be continuation marks in column 72.

4. You need to assemble, perform local modification on the RPIBLLNK build list, and build the RACFLINK LOADLIB. Follow the instructions in "Assemble a File, Modify a Build List, and Build a Library" on page 159, using the following substitution values:
   - For *fn* use **ICHRRCDE**
   - For *nnnn* use **0002**
   - For *blist* use **RPIBLLNK**
   - For *memname* use **ICHRRCDE**
5. Activate the class you defined (SETROPTS CLASSACT(your-class-name))
6. If you have RACROUTE applications that use a new installation-defined class, see "Using RACROUTE and Installation-Defined Classes" on page 29.

Changing an Installation-Defined Class

Changing certain fields of a class entry require extra attention. If you change the ID value for an existing class, you may get misleading messages from IRRUT200. If you change the ID value in the class descriptor table but not in the existing profiles, an incorrect profile may be associated with a specified class. If you change the POSIT value, follow up with the SETROPTS LIST command. As several classes may share the same POSIT value, changing the POSIT value may deactivate classes previously active and vice versa. If you change the MAXLNTH value, you may have to change some of your applications that invoke RACROUTE.

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

Follow this procedure:

1. Log on to the MAINT740 user ID.

2. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

For *compname*, use:

RACF
For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

3. Modify the ICHRRCDE ASSEMBLE program on the local modification 2C2 disk. You can modify a class entry by changing the macro statements for the class.
4. You need to assemble, perform local modification on the RPIBLLNK build list, and build the RACFLINK LOADLIB. Follow the instructions in "Assemble a File, Modify a Build List, and Build a Library" on page 159, using the following substitution values:
   - For *fn* use **ICHRRCDE**
   - For *nnnn* use **0002**
   - For *blist* use **RPIBLLNK**
   - For *memname* use **ICHRRCDE**
5. Activate the class you defined (SETROPTS CLASSACT(your-class-name))
6. If you have RACROUTE applications that use a new installation-defined class, see "Using RACROUTE and Installation-Defined Classes" on page 29.

Deleting an Installation-Defined Class

You can delete a class entry by removing the macro statements for the class from ICHRRCDE. Follow this procedure:

1. Log on to the MAINT740 user ID.
2. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

For *compname*, use:

RACF
For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

3. Delete the class entry by removing the macro statements for the class from the ICHRRCDE ASSEMBLE program on the local modification 2C2 disk.
4. You need to assemble, perform local modification on the RPIBLLNK build list, and build the RACFLINK LOADLIB. Follow the instructions in "Assemble a File, Modify a Build List, and Build a Library" on page 159, using the following substitution values:
   - For *fn* use **ICHRRCDE**

- For *nnnn* use **0002**
- For *blist* use **RPIBLLNK**
- For *memname* use **ICHRRCDE**

For the deletion to take effect:

- Re-IPL each RACF service machine.

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

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

To illustrate: You may 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 RACF router table contains one or more entries for each entry in IBM-supplied portion of the class descriptor table. It also contains entries for DATASET and USER.

If an entry is added to the class descriptor table, a corresponding entry should be added to the RACF router table (using the ICHRFRTB macro).

The router table consists of the ICHRFR0X and ICHRFR01 modules described below.

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

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

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

You may 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.

In addition to router table entries required by your application, you need an entry with a blank requestor (REQSTOR) and subsystem (SUBSYS) for RACF to use.

Adding an Entry to the RACF Router Table

ICHRFR01 is the name of the installation-defined RACF router table.

To add an entry, you need to create an assembler program, ICHRFR01 ASSEMBLE, on the local modification 2C2 disk and build a library.

Provided below is a sample of code to be added to ICHRFR01.

```
ICHRFR01 CSECT
$USRCLS  ICHRFRTB CLASS=$USRCLS,ACTION=RACF
ENDTAB   ICHRFRTB TYPE=END
         END ICHRFR01
```

Ensure that the last entry of ICHRFR01 has TYPE=END.

Follow the instructions in “Build or Link-Edit a Library” on page 166, using the following substitution values:

- For *fn* use **ICHRFR01**

- For *blist* use **RPIBLLNK**
- For *memname* use **ICHRFR01**

For information on coding the ICHRFRTB macro, see *z/VM: RACF Security Server Macros and Interfaces*.

Using RACROUTE and Installation-Defined Classes

Each user entering RACROUTE invocations with installation-defined classes needs access to an RPICDE LOADLIB file. To create the RPICDE LOADLIB:

1. Link to the RACF service machine's 305 disk (or log on to the RACMAINT user ID).
2. Create a link-edit file (called, for example, CDE TEXT) that contains the following statements:

   ```
   INCLUDE DD1
   NAME ICHRRCDE(R)
   ```

   Note: The INCLUDE and NAME statements must begin in column 2.
3. Perform FILEDEFs to prepare for the link-edit:

   ```
   FILEDEF DD1 DISK ICHRRCDE TXT00000 B
   FILEDEF SYSLMOD DISK RPICDE LOADLIB A (DSORG PO RECFM U BLOCK 13030 PERM
   ```
4. Perform a link-edit to create RPICDE LOADLIB:

   ```
   LKED CDE (LIST LET NCAL XREF PRINT MAP NOTERM DCBS LIBE RACFSYS SIZE 200K,200K
   ```

Each user needing to perform RACROUTE invocations that specify installation-defined classes must have access to this loadlib.

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) algorithm. The masking algorithm is the original algorithm provided with RACF. The RACF DES algorithm provides a higher level of security than the masking algorithm. 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.

The RACF encryption algorithms provide a high level of security because they support one-way encryption only; they do not support the reverse process. In addition, they do not store the passwords they use as the encryption keys. 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.

The RACF KDFAES Algorithm

The KDFAES algorithm is used to encrypt passwords and password phrases. 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 “Planning Considerations for Enabling KDFAES” on page 32 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

Two-step Method of User Verification

To move easily from a masked form to a DES form of authentication, RACF provides a two-step method of user verification. The method of user verification is used without an ICHDEX01 exit. To use this method, on systems prior to RACF function level 540, delete the ICHDEX01 exit that is shipped with RACF. On RACF function level 540 and later, the ICHDEX01 exit is disabled by default. Repeat this procedure for each system that shares the database.

Removing the exit causes a gradual migration from the masking algorithm to the RACF DES algorithm. The migration works as follows:

- For the RACF DES encoding function and the DES compare function:
  - When users (using the masked form of encryption) need to change their passwords, RACF DES treats the new, user-supplied passwords as encryption keys to transform the RACF user IDs into encoded forms that it stores on the RACF database.
  - Each time the users log on and enter their passwords, RACF again encrypts the user IDs using the passwords as the keys. RACF then compares the results with the encoded forms already stored on the database.
- For the masking compare function:
  - If RACF is unable to verify the users' passwords by means of the RACF DES compare function, it attempts to verify the users by comparing the results of the masking algorithm to the encoded forms of the passwords already stored on the database.

When converting from masking to RACF DES using the two-step method of checking, there is an extremely remote possibility that the *RACF DES-encoded* form of one user's password is identical to the *masked* form of another user's password. As long as your installation uses the two-step method of checking, your installation may have an exposure. To avoid this possibility, after all the users at your installation have been RACF DES-encoded using the two-step verification and conversion process, you should reactivate the ICHDEX01 exit and set the return code to 8. This return code always directs RACF to use only the RACF DES algorithm for logon checking.

Notes:

1. ICHDEX01 resides in RACFLPA LOADLIB.
2. If two or more systems share the RACF database, when you activate the RACF DES algorithm, activate RACF DES at the same time for all the systems that are sharing the RACF database.

For further information on ICHDEX01, see Chapter 6, “RACF Installation Exits,” on page 107.

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 that sets the return code to 4. See “Password-Encryption Exit” on page 132.

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 to perform your authentication algorithm, and set the return code accordingly. See “Password-Encryption Exit” on page 132.

PassTicket Authentication

The RACF secured signon function provides a *PassTicket* as an alternative to the RACF password. The RACF PassTicket is a *one-time-only* password that is generated by a requesting product or function. It is an alternative to the RACF password that removes the need to send RACF passwords across the network in clear text.

The PassTicket makes it possible to move the authentication of a mainframe application user ID from RACF to:

- Another authorized function running on the host system
- The workstation local area network (LAN) environment

If RACF authenticates a password field and determines that it is not the RACF password for the user ID, RACF performs a second authentication step to determine whether the password field is a valid PassTicket. See "How RACF Processes the Password or PassTicket" on page 32 for more information. See *z/VM: RACF Security Server Macros and Interfaces* for information on generating PassTickets.

How RACF Processes the Password or PassTicket

To validate a password or PassTicket, RACF:

1. Determines whether the value in the password field is the RACF password for the user ID.
   - If it is the RACF password, the validation is complete.
   - If it is not the RACF password, processing continues.
2. Determines whether a secured signon application profile has been defined for the application in the PTKTDATA class.
   - If a profile has not been defined, RACF sends a message to the user ID indicating that the password is not valid.
   - If the application is defined to the PTKTDATA class, processing continues.
3. Evaluates the value entered in the password field. The evaluation determines whether:
   - The value is a PassTicket consistent with this user ID, application, and time range.
   - It has been used previously on this computer system for this user ID, application, and time range.

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

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

Note:

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

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 33
- "Ensure that all systems sharing the RACF database are at the correct service level" on page 33
- "Determine if programs you have written are affected" on page 33
- "RACF exit considerations" on page 33
- "Performance and space considerations" on page 34

[1] GMT is also referred to as coordinated universal time (UTC).

- “Test the change” on page 35

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.

Ensure that all systems sharing the RACF database are at the correct service level

All z/VM systems that share the RACF database must be running at least z/VM V6.3 with APAR VM65719.

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=(dataaddr, 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 or masking 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.

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.

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 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:

- 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.

Note: You must have ECMODE on to run the report writer. This is because the ECMODE is necessary for executing certain instructions.

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

ICHRSMFI is an installation-replaceable, nonexecutable 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 1. 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 meaningful table provided
SORTTBLS	277(115)	2	SORT/MERGE alternate-sequence distribution-table size. (This parameter equals the actual number of nonblank 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 nonblank characters in the SORTDYN parameter field.)	Binary. Zero means no dynamic allocation by SORT/MERGE	14
SORTEQU	313(139)	8	SORT/MERGE preservation of input order for records with equal sort fields	EBCDIC (EQUALS or NOEQUALS	NOEQUALS)
SORTEQUS	321(141)	2	SORT/MERGE SORTEQU field size. (This parameter equals the actual number of nonblank characters in the SORTEQU parameter field.)	Binary (6 for EQUALS and 8 for NOEQUALS)	8
SORTDSN	323(143)	44	SORT/MERGE SORTLIB data set name. May be blanks if no SORTLIB is needed.	EBCDIC, left-justified, and padded with blanks	SYS1.SORTLIB
OUTSPA1	367(16F)	2	SYSPRINT primary-space allocation (in tracks)	Binary	50
OUTSPA2	369(171)	2	SYSPRINT secondary-space allocation (in tracks)	Binary	20
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	4096
WRKBLKSI	380(17C)	2	SORTIN block size	Binary	3256

Table 1. Format of ICHRSMFI (continued)

Name	Offsets DEC(HEX)	Length	Description	Format	Default
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

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

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

Setting the CP Disposition for Access Requests

Until you have RACF installed to your satisfaction, you might want CP to continue to make access decisions for some of your resources; for example, nodes, minidisks, and commands. (For the protected commands in VM, refer to *z/VM: RACF Security Server Security Administrator's Guide*.

The SYSSEC macro establishes a relationship between RACF's response to an access request and the final disposition of that request.

The defaults for SYSSEC parameters are shown in Table 2 on page 37.

You should be careful about changing the CP Disposition on minidisk relationships. Do not change it to Disallow Access. Resources are not defined when IBMUSER logs on after the initial IPL. If you disallow access, IBMUSER is not granted access to CMS minidisks that IBMUSER requires to initialize the RACF database.

See *z/VM: RACF Security Server Macros and Interfaces*. for a description of the SYSSEC macro.

Table 2. Initial Relationships between Access Decisions Made by RACF and Final Disposition by CP

RACF Response	CP Disposition
Access Permitted	Allow Access
Resource Undefined	Defer to CP
Access Denied	Disallow Access
Access denied, but warning mode is set for resource	Defer to CP

In the figure, if a user attempts to LINK to a minidisk that has not been defined to RACF, the request is deferred to VM. VM permits the user to link if the user has supplied a valid LINK password.

To update or change the SYSSEC macro invocation parameters in HCPRWA, put a local modification on to HCPRWA RPIBASE0. See "Applying local service and local modifications" in z/VM: Service Guide for the steps on how to apply a local modification to HCPRWA.

Suppressing Issuance of RACF Messages

Options in the SYSSEC macro allow you to suppress the issuance of RACF-defined error messages which result from unsuccessful authorization checks by RACF (messages issued by the VM operating system are still displayed. Messages can be suppressed for any combination of the resource classes VMMDISK, VMRDR, VMNODE, VMCMD and VMLAN.

The default is for messages to be displayed. To change the settings, you need to change the SYSSEC parameters in HCPRWA.

See *z/VM: RACF Security Server Macros and Interfaces*. for a description of the SYSSEC macro.

To update or change the SYSSEC macro invocation parameters in HCPRWA, put a local modification on to HCPRWA RPIBASE0. See "Applying local service and local modifications" in z/VM: Service Guide for the steps on how to apply a local modification to HCPRWA.

Defining Public Minidisks

Within the CP modules, you can define public minidisks (minidisks for public access). Defining public minidisks can improve performance because read access to them is automatically granted without calling the RACF service machine. To define a minidisk as public, use the GLBLDSK macro to define the minidisk in the global minidisk table. For information on the global minidisk table and how to identify minidisks that can be considered public minidisks, refer to *z/VM: RACF Security Server Macros and Interfaces*.

To update or change the GLBLDSK macro invocation parameters in HCPRWA, put a local modification on to HCPRWA RPIBASE0. See "Applying local service and local modifications" in z/VM: Service Guide for the steps on how to apply a local modification to HCPRWA.

Requiring Passwords for RACF Command Sessions

RACF does not require that users enter their passwords to establish RACF command sessions. However, if your installation wants its users to enter passwords for RACF command sessions, you must change the macro instruction statement in HCPRWA from:

- ```
  HCPRWA   RACFWA  CSECT=YES
  ```
- to
- ```
  HCPRWA   RACFWA  CSECT=YES,CMDPW=YES
  ```

To update or change HCPRWA, put a local modification on to HCPRWA RPIBASE0. See "Applying local service and local modifications" in *z/VM: Service Guide* for the steps to local mod HCPRWA.

For information on RACF command sessions, see *z/VM: RACF Security Server Command Language Reference*. Note that a logon password is always required, even if a password is not required for RACF command sessions.

Changing User IDs for RACF Service Machines

If you are using user IDs other than RACFVM and RACMAINT; for the RACF service machines, you need to update the RACSERV macro statements in HCPRWA. See *z/VM: RACF Security Server Macros and Interfaces* for details on the RACSERV macro.

To update the RACSERV macro statements in HCPRWA, put a local modification on to HCPRWA RPIBASE0. See "Applying local service and local modifications" in z/VM: Service Guide for the steps on how to apply a local modification to HCPRWA.

Note: The RACSERV statements must immediately follow the HCPRWATB entry definition label in HCPRWA.

Defining Multiple RACF Service Machines

It is strongly recommended that you install the RACF product and determine the overall system performance characteristics before you install multiple RACF service machines. This document contains information on the benefits of using multiple RACF service machines, how to determine if you need more than a single RACF service machine, and the procedure for installing multiple service machines.

If you plan to install multiple RACF service machines, IBM recommends that you make changes to the RACSERV invocations in HCPRWA as part of RACF installation, leaving the other multiple service machine installation tasks for a later time. This allows you to avoid a CP nucleus regeneration at that time.

To update the RACSERV macro statements in HCPRWA, put a local modification on to HCPRWA RPIBASE0. See "Applying local service and local modifications" in z/VM: Service Guide for the steps on how to apply a local modification to HCPRWA.

Note: The RACSERV statements must immediately follow the HCPRWATB entry definition label in HCPRWA.

Specifying the Value of the POSIX Constant NGROUPS_MAX

The POSIX constant NGROUPS_MAX defines the number of supplemental GIDs that are associated with a POSIX process for authorization. You specify the value of the NGROUPS_MAX constant on the ICHNGMAX macro. See *z/VM: RACF Security Server Macros and Interfaces* for a description of the ICHNGMAX macro. If a valid value for the NGROUPS_MAX constant is specified on the ICHNGMAX macro at initialization, RACF support for OpenExtensions for VM is activated.

We do not recommend that you specify a value for NGROUPS_MAX at install time unless you are sure that you want RACF support for OpenExtensions for VM activated at install time. *z/VM: RACF Security Server Security Administrator's Guide* contains a complete description of the steps required to activate the RACF support for OpenExtensions for VM. If you specify a value for NGROUPS_MAX at install time, make sure that you also perform the steps documented in *z/VM: RACF Security Server Security Administrator's Guide*.

Chapter 4. Operating Considerations Unique to z/VM

Understanding RACF Interaction with CP

The RACF modules residing in the VM control program provide the interaction between RACF and VM. By convention, the module names begin with the prefix HCP. RACF modules residing in VM CP are:

- HCPRPD
- HCPRPF
- HCPRPG
- HCPRPI
- HCPRPP
- HCPRPW
- HCPRWA

RACF includes UPDATE files for HCPRWA, HCPRPD, HCPRPF, HCPRPG, HCPRPI, HCPRPP, and HCPRPW with a filetype of RPIBASE0. During installation, the contents of these RACF UPDATE files replace the contents of the corresponding VM/CP ASSEMBLE files.

Note: An update file for HCPRWAC is also provided. This part is the Common Criteria compliant version of HCPRWA. If you intend to run a Common Criteria complaint configuration, see *z/VM: Secure Configuration Guide* .

Dynamic Parse

Dynamic parse is used by the RACF command processors to add, list, alter, or delete DFP, OVM, or any other nonbase-segment information.

Dynamic parse is started automatically by each RACF service machine during its initialization sequence.

If IBM makes changes to the dynamic-parse specification data (the PARSE FILE on RACF service machine's 305 disk) and the RACF database templates, the following procedure should be used:

1. Log on to the RACF service machine.
2. IPL the 490 minidisk.
3. Run the RACFCONV EXEC to update the templates in the RACF database.
4. Update or replace the PARSE FILE on the RACF service machine's 305 disk.
5. Issue RACSTART to resume operation.

Group Tree in Storage

Group tree in storage occurs automatically with installation and improves performance of users who have RACF group-SPECIAL, group-OPERATIONS, and group-AUDITOR attributes.

Database Considerations

RACF databases that reside on ECKD DASD can be shared with another z/VM system. Ensure that all z/VM configuration rules are followed so that z/VM honors RESERVE/RELEASE, which prevents RACF database corruption.

RACF databases can be shared with another z/VM system.. Ensure that all z/VM configuration rules are followed so that z/VM honors RESERVE/RELEASE, which prevents RACF database corruption.

© Copyright IBM Corp. 1993, 2025

You must add the V suffix when you define each database minidisk in the user directory. The V-suffix configuration is required for concurrent RACF database maintenance, even if the database is not shared with another z/VM system. If the RACF databases are not defined with the V suffix on the minidisk link mode (e.g. "MWV"), then RACF reports the following error and immediately logs off to prevent damage to the RACF databases:

```
CSTERP001E Minidisk ... was defined without the V suffix in the CP directory (e.g. MWV).
```

See “Sharing RACF Databases with Another z/VM System” on page 43.

Notes:

1. RACF database sharing with z/OS is not supported on z/VM 7.3 and later versions. After the RACF database template is upgraded for z/VM 7.3 by using the RACFCONV utility, database sharing is not supported. Database sharing is supported on z/VM 7.2 and earlier versions and is documented in z/VM 7.2 and earlier versions publications.
2. A RACF database residing on FBA (SCSI) DASD cannot be shared.

When sharing databases, consider the following options when defining DASD:

- If sharing between real systems (for example, two separate processors):
  - Assign the entire volume using either the CP directory DEDICATE statement or full-pack MDISK statement.

    In deciding which statement to use be aware that the DEDICATE statement lets only one user access the disk drive through that **cuu** address, whereas the full-pack MDISK statement lets the disk be shared among virtual machines. A full-pack minidisk is a single minidisk allocated on an entire DASD volume encompassing all the primary (or addressable) tracks of the volume.

    If using an MDISK statement for the RACF database, it must be set up with MWV (for shared multiwrite mode). Virtual reserve/release must be enabled for any guest virtual machine reserve to be accepted, even if only one guest (such as RACFVM), is using the device. Virtual reserve/release allows reserve/release CCWs to be issued by the guest. Real reserve/release allows CCWs to reach the device. CP issues the reserve/release CCWs if both virtual reserve/release and real reserve/release are enabled.
  - Define the DASD as shareable to CP by coding it as shared in IOCP and coding the SHARED operand on the RDEVICE System Configuration file statement. Setting the SHARED operand to YES applies only to sharing full-pack minidisks between multiple real systems. (You can make the DASD sharable in between system IPLs by using the CP SET RDEVICE command with the SHARED YES operand.)
- If sharing between virtual systems (for example, virtual and second-level guest systems):

  This method is for sharing a minidisk between virtual systems on the same real system and not for sharing with another real system. z/VM CP simulates reserve/release if virtual reserve/release is enabled and real reserve/release is not. Virtual reserve/release allows reserve/release CCWs to be issued by the guest.
  - The MDISK Statement in the directory should specify an access mode of MWV (for shared multiwrite mode). The V suffix tells CP to support virtual reserve/release on I/O operations to the minidisk.
  - Specify that the DASD will not be shared with another operating system. The default setting of the SHARED operand on the RDEVICE System Configuration file statement, which is SHARED NO, enables this configuration setting. A NO setting means this volume cannot be shared with an operating system running on another processor. (CP overhead is greater when you specify SHARED=YES as it directs CP to pass real reserve/release CCWS to the Real Device. It is not necessary for "Virtual-Only" sharing).
- If sharing between real and virtual systems (for example, two separate processors with at least one second-level guest):

  This method of sharing DASD, concurrent virtual and real reserve/release, combines virtual reserve/release and real reserve/release. It allows DASD to be shared among multiple virtual machines and operating systems running on other processors.

Concurrent virtual and real reserve/release involves the use of full-pack minidisks. The requirements are the same as when sharing between real systems except that DEDICATE is not an option. The DASD must be defined to CP as shareable and the MDISK directory statement must specify an access mode of MWV.

To summarize:

- Use a full-pack minidisk (includes cylinder zero)
- The MDISK directory statement must use the V statement (for example, MWV)
- The System Configuration file requires RDEVICE statement with SHARED YES specified.

- If sharing with another system, ensure that the full pack minidisk on which the RACF database resides is **NOT** on a CSE formatted volume.
- MDC (minidisk caching) should be OFF for RACF database DASDs. The MDC feature is incompatible with DASD sharing in any form.
- z/VM CP and the directory program do not completely prevent you from defining minidisks that overlap. Overlap can defeat the integrity of link access modes and RESERVE/RELEASE serialization. Therefore, ensure that you validate the MDISK definitions for the RACF database volumes; overlaps must match completely.

Sharing RACF Databases with Another z/VM System

The z/VM restrictions and requirements for sharing DASD must be met. Information concerning z/VM requirements can be found in the z/VM library and should be reviewed for planning purposes:

If you are sharing with a z/VM system, see *z/VM: Running Guest Operating Systems*.

Decide where the RACF databases and libraries will be located, and whether there will be a single database or multiple databases.

Shared DASD Considerations

Whatever the benefits of DASD sharing you'd like to bring to your installation, consider the cost of additional complexity and any performance implications. Elements to consider and plan for include:

- System design in terms of DASD mapping
- Resource and load balancing
- Recovery and restart
- Operational control of the multiprocessor environment
- Programming considerations for user resource protection
- Data integrity

Sharing RACF Databases in a z/VM Single System Image Cluster

When RACF is installed in a z/VM single system image (SSI) environment, it is mandatory that the RACF database is shared. To ensure database integrity the following requirements must be met.

- The RACF database DASD must reside on ECKD DASD.
- The RACF database DASD must be defined as shared in the I/O configuration. See *z/VM: I/O Configuration* for information on how to define a device as shared.
- Both the primary RACF database (device 200) and the backup database (device 300) must be defined on full-pack minidisks. It is also required that these devices have virtual reserve/release enabled. The following example shows how to define the RACF database disks in the CP user directory entry for the RACFVM server.

```
MDISK 200 3390 0 END RACF MWV READ WRITE MULTIPLE
MDISK 300 3390 0 END RACFBK MWV READ WRITE MULTIPLE
```

The mode suffix of “V” on the MDISK statements tells CP to use virtual reserve/release support in the I/O operations for the full-pack minidisk.

In order to associate the RACF database and its backup with a `0..END` defined fullpack volume, it must previously be defined to the list of real devices associated with user volumes (USER_VOLUME_RDEV) in the SYSTEM CONFIG file:

```
USER_VOLUME_RDEV:
USER_VOLUME_RDEV RACF RDEV rdev
USER_VOLUME_RDEV RACFBK RDEV rdev
```

Also, update any LINKs to RACFVM 200 and RACFVM 300 minidisks that are MR to MW. For example, the CP user directory entry for the RACMAINT test server.

```
 LINK RACFVM 200 200 MW
 LINK RACFVM 300 300 MW
```

If you are following the recommended practice of using the same real device numbers across LPARs to reference DASD, the MDISK statements for the RACF database disks can be placed in the identity entry for the RACF server. If the real device numbers are not the same across LPARs, the MDISK statements must be placed in the relevant sub-configuration entries.

- To define the RACF database DASD to CP as devices that can be shared concurrently between real systems, you must add the RDEVICE statements (as shown in the following example) to the CP system configuration file (SYSTEM CONFIG).

```
RDEVICE rdev TYPE DASD SHARED YES    /* RACFVM Primary Database */
RDEVICE rdev TYPE DASD SHARED YES    /* RACFVM Backup Database */
```

It is a requirement that CP does not cache data on the RACF database disks in the minidisk cache. Minidisk cache (MDC) is turned off as a result of specifying the DASD as shared in the system configuration file.

It is no longer recommended that you use the DEVNO operand of the MDISK directory statement to define the DASD as fullpack minidisks. If you are supporting an environment where DEVNO is in use, understand that the MDISK statements previously shown will written in a different format (owing to differences in syntax). Additionally, it is a requirement that the DASD volumes on which the RACF database full-pack minidisks reside cannot be listed on the USER_VOLUME_LIST statement in the SYSTEM CONFIG file when the DEVNO operand is used on the MDISK directory statement to define the full-pack minidisks.

Note: In a GDPS® environment, the DEVNO operand on the MDISK statement cannot be used because DASD real devices cannot be hardcoded in the user directory. Instead, RDEVICE statements for the real devices that will contain the RACF primary and backup database minidisks for the GDPS primary and secondary sites must be added to the Site 1 and Site 2 system configuration files. Additionally, the DASD volumes for both real devices must be added to the USER_VOLUME_RDEV statement in each of the Site 1 and Site 2 system configuration files:

```
USER_VOLUME_RDEV volid_a RDEV rdev
USER_VOLUME_RDEV volid_b RDEV rdev
```

(where *rdev* defines the unique real device addresses of the PPRC primary DASD on Site 1 and Site 2.) See *z/VM: CP Planning and Administration* for more information.

Then, both the RACF primary and backup database full-pack minidisks are defined in the CP directory entry for the RACFVM server as follows:

```
MDISK 200 3390 0 END volid_a MWV READ WRITE MULTI
MDISK 300 3390 0 END volid_b MWV READ WRITE MULTI
```

Initializing a RACF database minidisk residing on FBA DASD

When a new minidisk is to be used as a RACF/VM database minidisk, either as a primary RACF database (RACF) or backup RACF database (RACFBK) or as a RACF work minidisk, the minidisk must be prepared first. To prepare a minidisk residing on FBA DASD to be a RACF database minidisk, do the following:

1. Use the CMS FORMAT command to format the minidisk with a blocksize of 4096:

   ```
   FORMAT vdev fm (BLK 4096
   ```

   Where:

   vdev
   is the virtual device address of the minidisk.

   fm
   is the filemode at which the virtual device is accessed,

2. Use the RACINITD command to initialize the database minidisk:

   ```
   RACINITD
   ```

   Enter the virtual device address for CUU.

RACF Database on Full-pack Minidisk

When the RACF database is placed on a full-pack minidisk it is desirable to limit the size of the database on the full-pack and not use the entire allocation of the volume. Before moving your database to a full-pack you must decide whether you want to retain the current size of the database (as defined in the CP user directory), or whether you want to increase the size of your database when it is moved.

Moving from Minidisk to Full-pack while Keeping the Current Database Allocation

The following procedure assumes one primary and one backup database disk. If your database is split then you will have to repeat the process for the other disks remembering to use the relevant primary, backup, and work disk addresses and device labels.

The initial configuration has the primary and backup database disks defined on 17 cylinder minidisks. The aim is to move these databases to full-pack minidisks and continue to only have the database as 17 cylinders in size on the full-pack minidisk.

This procedure can be carried out by a user who has read access to the RACFVM 200 and 300 disks, and has write access to the new full-pack minidisks.

1. Copy the primary database to our new primary full-pack. In this example, our full-pack is device 9200.

   ```
   DDR
   SYSPRINT CONS
   IN 200 3390
   OUT 9200 3390
   COPY 0 TO 16
   YES
   Enter
   YES
   Enter
   ```

2. Copy the backup database to our new backup full-pack. In this example, our full-pack is device 9300.

   ```
   DDR
   SYSPRINT CONS
   IN 300 3390
   OUT 9300 3390
   COPY 0 to 16
   YES
   Enter
   YES
   Enter
   ```

Note: If you are performing the above steps as part of a migration process you must now upgrade the RACF Database Templates using RACFCONV. See "RACF Database-Initialization Utility Program (IRRMIN00)" on page 69 for further information. Also, it is recommended that RACUT200 be used to verify the primary and backup databases, before running RACFCONV. If there are errors, they must be corrected before running RACFCONV.

You can now use these new devices as your primary and backup database disks.

Note: It is highly recommended to keep a current copy of your primary RACF database distinct from your primary and backup volumes in case it is needed for recovery.

Moving from Minidisk to Full-pack while Increasing the Database Allocation

The following procedure assumes one primary and one backup database disk. If your database is split then you will have to repeat the process for the other disks remembering to use the relevant primary, backup, and work disk addresses and device labels.

The initial configuration has the primary and backup database disks defined on 17 cylinder minidisks. The aim is to move these databases to full-pack minidisks but have the primary and backup database be 100 cylinders each and not use the full size of the volume.

The RACMAINT user ID can be used to perform the following task. If this ID is currently in use, you can use any user that can:

- Link to the RACFVM 305 disk and access it
- Link to the primary database disks in write mode

In addition to the above requirements you must define the following minidisks to the user ID being used:

- A 100 cylinder backup database disk, virtual address 300
- A 100 cylinder work disk, virtual address 400

Sign on to RACMAINT (or your chosen user ID) and perform the following steps:

1. OS format your new backup and work disks by running RACDSF using the values shown in the table below.

   Note: The default labels of RACFBK and RACF can be changed on the RACDSF panel if required. This may be necessary if you have specific installation naming standards.

   Table 3. RACDSF values

	VADDR	Label
Backup Disks	300	RACFBK
Work Disks	400	RACF

   Press PF2 to OS-format both disks.
2. Allocate the new size backup dataset by running RACALLOC. Specify 300 as the disk on which you wish to allocate the database. Ensure that RACF.BACKUP is allocated.
3. Allocate the new size primary database on the work disk by running RACALLOC. Specify 400 as the disk on which you wish to allocate the database. Ensure that RACF.DATASET is allocated.
4. Expand the primary database to the size required.

```
RACUT400
Enter
copy
200
400
check that the input and output databases / devices are correct.
YES
CONT
specify your parameters as required (for example LOCKINPUT, NOLOCKINPUT, etc).
Note that if you specify LOCKINPUT you will need to run RACUT400 again once
the procedure is complete specifying UNLOCKINPUT and 200 as the input and
```

```
output device address.
END
Review the output UT400 OUTPUT A.
```

5. Copy the new size primary to the new size backup.

```
RACUT200
Enter
NO
400
300
Check that the input and output dataset names are correct.
YES
```

6. Copy the resized primary database to our new primary full-pack. In this example, our full-pack is device 9200.

```
DDR
SYSPRINT CONS
IN 400 3390
OUT 9200 3390
COPY 0 TO 99
YES
Enter
YES
Enter
```

7. Copy the resized backup database to our new backup full-pack. In this example, our full-pack is device 9300.

```
DDR
SYSPRINT CONS
IN 300 3390
OUT 9300 3390
COPY 0 to 99
YES
Enter
Yes
Enter
```

You can now use these new devices as your primary and backup database disks.

MDISK 200 3390 DEVNO 9200 MWV READ WRITE MULTIPLE
MDISK 300 3390 DEVNO 9300 MWV READ WRITE MULTIPLE

Note:

1. Any changes made to the original database since this procedure was carried out will not be reflected in the new database. If you need to refresh the new database you can use RACUT400 to copy from the current primary to the new full-pack primary. The size of the database on the full-pack will not be changed since we are now using the utilities and not using DDR to copy the volume 1 label and data.
2. If you are performing the above steps as part of a migration process you must now upgrade the RACF Database Templates using RACFCONV. See “RACF Database-Initialization Utility Program (IRRMIN00)” on page 69 for further information. Also, it is recommended that RACUT200 be used to verify the primary and backup databases, before running RACFCONV. If there are errors, they must be corrected before running RACFCONV.
3. It is highly recommended to keep a current copy of your primary RACF database distinct from your primary and backup volumes in case it is needed for recovery.

Moving a Full-pack Minidisk to an Equally Large or Larger DASD

The instructions that follow can be used to move RACF database full-pack minidisks to equally large or larger DASD.

Notes:

- This procedure can only be applied if the following are true:
  - The disk architecture does not change.

  - The database size should not be resized.
- RACFVM will not be available during this change, and the system will be in hard fail mode. Only very basic actions can be taken that need authorization. The USER DIRECT will be in effect.
- Ensure that you have a current copy of the USER DIRECT accessible outside of the z/VM system.
- Keep logged on at one userid that can change and activate the USER DIRECT until all steps are completed.
- Keep logged on at one userid that can force and XAUTOLOG RACFVM on all SSI members.

Assumptions

These instructions assume the following:

- Old primary DASD RDEV and volid: 500 , RACF
- Old backup DASD RDEV and volid: 501 , RACFBK
- New primary DASD RDEV and volid: 700 , RACFBK
- New backup DASD RDEV and volid: 701 , RACNBK

Preparation

To prepare for this procedure, follow these steps:

1. Verify the database integrity. To do so, see https://www.ibm.com/support/pages/node/7114056.
2. Validate that the current RACFVM user config is configured one of these two ways:

   For a shared database, SSI:

   ```
   USER_VOLUME_RDEV:
   USER_VOLUME_RDEV RACF RDEV 500
   USER_VOLUME_RDEV RACFBK RDEV 501

   MDISK 200 3390 0 END RACF MWV READ WRITE MULTIPLE
   MDISK 300 3390 0 END RACFBK MWV READ WRITE MULTIPLE
   ```

   For an isolated database:

   ```
   USER_VOLUME_RDEV:
   USER_VOLUME_RDEV RACF RDEV 500
   USER_VOLUME_RDEV RACFBK RDEV 501

   MDISK 200 3390 0 END RACF MW READ WRITE MULTIPLE

   MDISK 300 3390 0 END RACFBK MW READ WRITE MULTIPLE
   ```

General setup instructions

To proceed with moving the minidisk, follow these steps:

1. Define new full-packed minidisks on 3390-9 DASDs as temporary vdev on one system.

   Assumption: In the examples used in these instructions, 0700 and 0701 are the new DASDs.

   In RACFVM USER DIRECT:

   a. Add this statement below the MDISK 200 statement:

   ```
   MDISK 220 3390 0 END RACNEW MW READ WRITE MULTIPLE
   ```

   b. Add this statement below the MDISK 300 statement:

   ```
   MDISK 320 3390 0 END RACNBK MW READ WRITE MULTIPLE
   ```

2. Activate USER DIRECT via directxa or an appropriate DIRMAINT command.
3. Logon to one RACFVM guest and IPL CMS:

```
IPL CMS
```

4. To shut down all other RACFVM guests that are sharing the database, perform this on each SSI member, via the operator:

```
FORCE RACFVM
```

5. Validate that the number of cylinders on the 200 disk is less than or equal to the number of cylinders on the 220 disk and that the number of cylinders on the 300 disk is less than or equal to the number of cylinders on the 320 disk, as shown in this example:

```
Q V DASD
...
DASD 0200 3390 RACF   R/W       3339 CYL ON DASD  0500 SUBCHANNEL = 0008
DASD 0220 3390 RACNEW R/W      10017 CYL ON DASD  0700 SUBCHANNEL = 0009
DASD 0300 3390 RACFBK R/W       3339 CYL ON DASD  0501 SUBCHANNEL = 000A
DASD 0320 3390 RACNBK R/W      10017 CYL ON DASD  0701 SUBCHANNEL = 000B
...
```

6. Use DDR to copy the 200 disk to the 220 disk and the 300 disk to the 320 disk, as shown in the following example.

   Note: RC 4 can be expected.

```
DDR
z/VM DASD DUMP/RESTORE PROGRAM
ENTER:
input 200 3390
ENTER:
output 220 3390
ENTER:
copy 0 3338
HCPDDR711D VOLID READ IS RACF
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
yes
ENTER NEXT EXTENT OR NULL LINE
ENTER:
HCPDDR716D NO VOL1 LABEL FOUND
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
yes
COPYING   RACF
END OF COPY
ENTER:
END OF JOB
Ready(00004); T=0.01/0.01

DDR
z/VM DASD DUMP/RESTORE PROGRAM
ENTER:
input 300 3390
ENTER:
output 320 3390
ENTER:
copy 0 3338
HCPDDR711D VOLID READ IS RACFBK
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
yes
ENTER NEXT EXTENT OR NULL LINE
ENTER:
HCPDDR716D NO VOL1 LABEL FOUND
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
yes
COPYING   RACFBK
END OF COPY
ENTER:
END OF JOB
Ready(00004); T=0.01/0.01
```

7. Logoff RACFVM.
8. Update USER DIRECT statements:
   a. Remove old RACFVM 200 and 300 disks from USER DIRECT.
   b. Change RACNEW and RACNBK to be the new 200 and 300 disks.
   c. If running in SSI, make sure that MWV is specified, as shown in this example:

```
USER_VOLUME_RDEV:
USER_VOLUME_RDEV RACF RDEV 700
USER_VOLUME_RDEV RACFBK RDEV 701
MDISK 200 3390 0 END RACF MWV READ WRITE MULTIPLE (Was 700)
MDISK 300 3390 0 END RACFBK MWV READ WRITE MULTIPLE (was 701)
```

   Or

   If running isolated, specify MW instead, as shown in this example:

```
USER_VOLUME_RDEV:
USER_VOLUME_RDEV RACF RDEV 700
USER_VOLUME_RDEV RACFBK RDEV 701
MDISK 200 3390 0 END RACF MW READ WRITE MULTIPLE (was 700)
MDISK 300 3390 0 END RACFBK MW READ WRITE MULTIPLE (was 701)
```

   d. If the MDISKs are configured in identities, update all identities, depending on the environment.
9. Activate USER DIRECT via directxa or an appropriate DIRMAINT command.
10. Start all RACFVM guests again:

```
XAUTOLOG RACFVM
```

Moving User IDs and Minidisks between Systems

When it is necessary to move a user ID or minidisk from one system to another, you can use the following procedures.

Moving User IDs

To move a user ID from one system to another:

- Execute RACUT100 to invoke the cross-reference utility, IRRUT100.

  The EXEC finds all the resources to which the user ID has access and all the groups to which the user ID belongs. The security administrator can take the output from RACUT100 and issue an RLIST for each resource to which the user has access. The security administrator then issues commands on the target system to give the user the equivalent of what the user had on the initial system.

  The security administrator can issue a LISTUSER command for each user ID and issue the appropriate commands on the target system to reflect the groups to which the user belonged on the original system.

  The drawback to using this approach is that RACUT100 must be run in a deactivated RACF service machine.
- Specify the user ID parameter on the SEARCH command.

  If you issue a SEARCH for each class in which the user has resources, it provides a listing of all the resources in that class to which the user has access.

  With this approach, the installation would issue the RLIST command for each resource to find out what authority the user has to it. It could next issue a PERMIT command on the target system to give the user the same access to each resource.

  Next, the security administrator issues a LISTUSER command for the user ID. With the information on the groups the user ID is connected to, issue the appropriate commands on the target system to reflect the groups to which the user belonged on the original system.

Note: The SEARCH command is CPU-intensive. It executes on the RACFVM service machine and can adversely affect performance. These procedures should be performed when there are few or no users on the system.

Moving Minidisks

When you move a minidisk from one system to another, everyone who had access to that minidisk loses it. Thus, you must structure your PERMIT commands on the target system to reflect what existed on the original system. You could use either of the methods described for moving user IDs to determine the users that have access to a minidisk and what that access level is.

Note: If your installation does not use dual registration, you need to ensure that you create a RACF profile to protect the minidisk.

Renaming a User

If you want to change a user's user ID, you have to perform the same steps as those described in "Moving User IDs and Minidisks between Systems" on page 50. You must ascertain all the resources to which the user has access, and all the groups to which the user belongs, and then issue the appropriate commands to re-create the user's environment.

After replicating the user's environment with the new user ID, you can delete the old user ID. See *z/VM: RACF Security Server Security Administrator's Guide* for an example of deletion.

Modifying Applications to Use the LOGON BY Function

The RACF LOGON BY function allows authorized users to logon to a shared user ID using their own password. The shared user ID must be defined to RACF as shared. Users authorized to use a shared user ID are called **surrogate** users,

The RACF LOGON BY function uses:

- The BY option of the LOGON command to specify the surrogate user
- The SURROGAT class to perform authorization checks for logons to shared user IDs

RACF allows one user ID to logon to a shared user ID if that user has at least READ access to the SURROGAT profile named LOGONBY.*shared_userid*.

You might need to modify applications that use RACROUTE to perform third party authorization requests. To preserve auditability, the application should issue Diagnose 26C subcode 4 to obtain the user ID that is currently logged onto the user ID for whom the RACROUTE is being issued. For more information, see *z/VM: CP Programming Services*.

If a surrogate user ID exists, it should be used in existing surrogate user ID support provided by the STOKEN and SUSERID keywords of the RACROUTE macro. For more information, see *z/VM: Security Server RACROUTE Macro Reference*.

Some applications prompt a user to enter a password as a means of authenticating the user ID before performing a requested function. If an application is invoked from a shared user ID, it may be desirable to validate the surrogate user ID and password rather than requiring the surrogate user to know the shared user ID's password. To do this, the application can:

1. Issue the z/VM Diagnose 26C subcode 4 to check if the requesting user ID is being shared. If it is being shared, Diagnose 26C subcode 4 returns the user ID that is currently logged on to it.
2. Pass the user ID to RACF using whatever interface the application currently uses (for example, Diagnose A0 subcode 4 or RACROUTE).

For details on Diagnose 26C subcode 4, see *z/VM: CP Programming Services*.

Using the GLBLDSK Macro

Use the GLBLDSK macro to create a global minidisk table for your installation's public minidisks. For a complete description of GLBLDSK, see *z/VM: RACF Security Server Macros and Interfaces*.

Along with the security administrator, you should identify the public minidisks for the installation. Use the following guidelines to determine public minidisks:

- They have no installation-sensitive data on them
- They are linked in R or RR mode
- All users are authorized to read the data on them
- They require no auditing

  Note: If ACIGROUPs are used on your system, the minidisk may be defined as public within the scope of your group.

Consider the following candidates for public minidisks:

- MAINT 190 system disk (CMS S disk)
- MAINT 19E system disk extension (CMS Y disk)
- ISPF minidisks
- OfficeVision minidisks
- Local system extension disks
- Tools minidisks

To use a global minidisk table, follow this procedure:

1. Create an update file for HCPRWA to specify the public minidisks.
2. Assemble HCPRWA.
3. Regenerate the CP nucleus.

For instructions on performing this local modification, see *z/VM: Service Guide*.

A minidisk may appear in the global minidisk table more than once. The entire table is scanned until a match is found or the end of the table is reached. Place the most frequently linked public minidisks at the top of the table.

It is recommended that for all minidisks in the global minidisk table you create a similar minidisk profile. Such a "matched pair" approach can help ensure the continuation of protection if your global minidisk table is changed in the future and GLBLDSK entries are removed. For example, create a profile with this command:

```
RDEFINE  VMMDISK  MAINT.190  UACC(READ)
```

This will allow all users to link in R or RR mode to the 190 minidisk of MAINT. The GLBLDSK entry doing the same thing would be:

```
GLBLDSK USERID=MAINT,VADDR=190
```

If someone is attempting WRITE access to a public minidisk, the global minidisk table is not searched. In this case, the RACF service machine is called for authorization checking.

Using the SFSAUTOACCESS Option

The SFSAUTOACCESS option allows the RACROUTE REQUEST=AUTH interface running in the SFS file pool server to grant access to files or directories automatically whenever users access their own SFS files or directories.

An access level is associated with the automatic access, similar to the RACF global access checking table. The access level specifies the highest access level allowed without calling the RACF service machine. If

a RACROUTE REQUEST=AUTH call specifies an access level higher than the access level specified with SFSAUTOACCESS, the request is sent to the RACF service machine for processing.

The RACF SERVMACH file contains two records:

- Record one contains the RACF service machine ID that receives RACROUTE requests:

```
Service Machine ID - RACFVM
```

- Record two contains the access level for automatic access to each user's SFS files and directories:

```
SFSAUTOACCESS=NONE
```

  The possible contents for record two are:

  SFSAUTOACCESS=NONE
  No automatic access is granted. All RACROUTE requests are sent to the RACF service machine. This is the default. SFSAUTOACCESS=NONE is also in effect if:

  – The syntax of the second record is incorrect

  – The second record is missing

  SFSAUTOACCESS=READ
  Gives all users READ access to their SFS files or directories without a call to the RACF service machine

  SFSAUTOACCESS=UPDATE
  Gives all users READ or UPDATE access to their SFS files or directories without a call to the RACF service machine

  SFSAUTOACCESS=CONTROL
  Gives all users READ, UPDATE, or CONTROL access to their SFS files or directories without a call to the RACF service machine

  SFSAUTOACCESS=ALTER
  Gives all users READ, UPDATE, CONTROL, or ALTER access to their SFS files or directories without a call to the RACF service machine. This results in the best performance.

Restriction:

When you use the SFSAUTOACCESS option:

- Access applies only to RACROUTE REQUEST=AUTH requests for the FILE and DIRECTRY classes.
- Access is granted only when the user ID attempting to access the resource matches the second qualifier of the file or directory being accessed. For example, user SIVLE can edit his PROFILE XEDIT in directory RESEARCH:SIVLE.HOUNDOG.
- When access isgranted according to the SFSAUTOACCESS value, and a profile exists for the resource being accessed, the profile is bypassed because the RACF secvice machine is not called. Therefore, the options in the profile are ignored.

The SFSAUTOACCESS option should *not* be used in the following circumstances:

- Security level, security category, or security label protection is being used in FILE and DIRECTRY profiles
- Auditing of any kind is desired for users accessing their own SFS files or directories

You can limit the SFSAUTOACCESS option to certain SFS file pool servers. For example, an installation could have two SFS file pool servers. File pool server SFS1 could have a copy of RACF SERVMACH containing SFSAUTOACCESS=UPDATE, and file pool server SFS2 could have a copy without the second record in RACF SERVMACH. Users enrolled in file pool SFS1 would get automatic read and write access to their files and directories in file pool SFS1 without calls to the RACF service machine. Users accessing their files and directories in file pool SFS2 would not get the same performance advantage.

Message Support

RACF simulates OS multiple-console support by using the z/VM WNG, SMSG, and MSGNOH commands.

Note: If you use MSG, the issuing user ID is printed out with the message.

The simulation uses a routing table (CSTCONS) to send console messages to z/VM user IDs. RACF provides the routing codes.

The Message-Routing Table

The CSTCONS routing table example shown in Figure 5 on page 54 contains the following:

- The user IDs that should receive the text of WTO/WTOR SVCs
- The command to be used when sending the text of the WTO/WTOR SVCs; for example: MSG, WNG, or MSGNOH
- A list of the routing codes each user ID in the table should receive.

```
CSTCONS  CSECT           MCS TABLE OF USER IDS
* ENTRY FORMAT:
*
* |-------------------------------------------------------------------|
* | ROUTCODES   |       TARGET USER ID           |      MSG MODE      |
* |-------------------------------------------------------------------|
* 0             2                               10                  15
MCSR1    EQU   X'80'          MASTER CONSOLE
MCSR2    EQU   X'40'          MASTER CONSOLE INFORMATIONAL
MCSR3    EQU   X'20'          TAPE POOL
MCSR4    EQU   X'10'          DIRECT ACCESS POOL
MCSR5    EQU   X'08'          TAPE LIBRARY
MCSR6    EQU   X'04'          DISK LIBRARY
MCSR7    EQU   X'02'          UNIT RECORD POOL
MCSR8    EQU   X'01'          TELEPROCESSING POOL
MCSR9    EQU   X'80'          SYSTEM SECURITY
MCSR10   EQU   X'40'          SYSTEM/ERROR MAINTENANCE
MCSR11   EQU   X'20'          PROGRAMMER INFO
MCSR12   EQU   X'10'          EMULATOR INFORMATION
MCSR13   EQU   X'08'          USER ROUTING
MCSR14   EQU   X'04'          USER ROUTING
MCSR15   EQU   X'02'          USER ROUTING
MCSR16   EQU   X'01'          AUTHORIZATION FOR SMSG TO RACF SERVER
*
FF       EQU   255
         SPACE 3
MASTRCON DC    AL1(FF),AL1(FF-MCSR9-MCSR11),CL8'*',CL6'MSGNOH'
SECCON   DC    X'00',AL1(MCSR9),CL8'OPERATOR',CL6'MSGNOH'
SEC2CON  DC    X'00',AL1(MCSR16),CL8'RACFSMF',CL6'MSGNOH'
END      DC    XL2'00'
         END
```

Figure 5. Example of CSTCONS Routing Table

If your installation uses the CSTCONS routing table shipped with RACF, the RACF service machine processing the request receives messages with all routing codes, except security-routing codes (route code 9) that go to the OPERATOR, and Write to Programmer messages (route code 11), which are ignored.

In this file, OPERATOR indicates multiple user IDs; specifically:

- The user ID "OPERATOR"
- The current system operator, who is selected dynamically and can be determined via the **CP QUERY SYSOPER** command.

If you want a user ID other than OPERATOR to receive the messages with security-routing codes, replace OPERATOR with that user ID in the CSTCONS table; if you want another user ID (such as a system administrator) in addition to OPERATOR to receive the messages with security routing codes, add that user ID to the CSTCONS table without deleting OPERATOR. You must add any additional user IDs to the table after the MASTRCON statement and before the END statement.

Whatever user IDs you add to this file will receive those messages and will be able to respond to them by issuing SMSG as indicated in “Enhanced Operator-Message Support” on page 55. Keeping OPERATOR in this list allows both the user ID "OPERATOR" and the current system operator to get the messages as well as to issue **SMSG server-name 1 text** in response to the message prompts, where *server-name* is the user ID of the RACF service machine. RACFVM is the IBM-supplied default.

For example, you would enter this statement after the SECCON statement (that identifies the OPERATOR in the CSTCONS table in Figure 5 on page 54) and before the END statement:

```
THRDCON  DC  X'00', AL1(MCSR9), CL8'SYSDA', CL6'MSGNOH'
```

This entry in the table causes the system to route security messages to SYSDA in addition to routing them to the operator.

To update the table, edit and reassemble the source file CSTCONS ASSEMBLE. Follow the instructions in “Modify Full-Part ASSEMBLE and TEXT Files” on page 160, using the following substitution values.

- For *fn* use **CSTCONS**
- For *nnnn* use **0002**

Note:

1. If a message is sent to a user who is not logged on when the message is sent, the message is forwarded to the RACF service machine processing the request. The user ID (in parentheses) of the user for whom the message was intended precedes the message.
2. IBM recommends placing the security administrator in the CSTCONS table so that the security administrator can receive security-related messages and respond to RACF prompts.
3. If using multiple RACF service machines, use the MESSAGE command to identify the originating user ID.

Enhanced Operator-Message Support

This support provides users defined to RACF in the CSTCONS table the ability to review and respond to outstanding messages that are generated by a RACF service machine.

Specifying OPERATOR in this list allows both the user ID "OPERATOR" and the current system operator to get the messages as well as to issue **SMSG server-name 1 text** in response to the message prompts. Appointing a new system operator will remove the authorization to reply to outstanding requests from the prior system operator.

To query for outstanding messages from a RACF service machine, a user defined in the CSTCONS table enters the following command:

```
#CP SMSG server-name QMSG
```

where *server-name* is the user ID of the RACF service machine. RACFVM is the IBM-supplied default.

Note: There is only one blank between the name of a RACF service machine and `QMSG`.

The RACF service machine then transmits to the user a list of outstanding messages, containing the response number and the response text, using either MSG, MSGNOH, or WNG commands (as specified in the CSTCONS table). If you use the MSG command, all messages are prefixed with a RACF service machine name (RACFVM); the MSG command is required in a multiple RACF service machine environment to ensure that you can identify the originating service machine.

The user response is:

```
#CP SMSG server-name 1 text
```

where 1 is the appropriate response number, and *text* is the required text dictated in the prompt sent from the server-name of the user ID of the RACF service machine. RACFVM is the IBM-supplied default.

Using Multiple RACF Service Machines

With RACF, you can have more than one service machine. With multiple RACF service machines, the security workload can be distributed among those machines. When you log onto a system, you are assigned to the service machine with the smallest number of users currently assigned. This server handles all your security-related requests until you log off.

With proper planning and tailoring for a particular system configuration, multiple RACF service machines can provide improved throughput.

Benefits

- Virtual Storage

  Most security-related processing is performed within the RACF service machine. Adding more RACF service machines on systems that have large numbers of users increases the effective virtual storage. The security workload can be distributed among those machines to improve the throughput of security requests.
- Availability

  If one of the RACF service machines becomes disabled, the remaining service machines continue to provide security services.
- Improved throughput capability for security relevant requests

  Multiple RACF service machines can process more security-related requests than a single RACF service machine.

The cost involved with multiple RACF service machines is the operating system management of the virtual machine. Because the RACF database minidisks are shared between the multiple RACF service machines, part of the cost may be an increase in the I/O activity to the database minidisks.

Considerations for Using Multiple RACF Service Machines

The following considerations can help you determine if you want to use more than one RACF service machine:

- Applications that issue large number of RACF commands

  Consider dedicating a server to processing such applications using the RAC command. Multiple applications of this type can share one server, or a server can be dedicated to one application only.

  1. Code CPUSE=NO on the RACSERV macro for that RACF service machine. Refer to *z/VM: RACF Security Server Macros and Interfaces* for information on the RACSERV macro.
  2. Change the application to set the $RAC_SRV global variable to the user ID of the dedicated RACF service machine.
- RACROUTE service

  If you have products or applications that employ the full function RACROUTE interface you should consider dedicating a RACF service machine to process RACROUTE requests. See "Dedicating RACF Service Machines for RACROUTE Request Processing" on page 57 if you choose this option.
- Discrete profiles for minidisks with large access lists

  If you use RACF to provide access control for minidisks, and the majority of the minidisks have been defined with discrete profiles and contain large access lists, consider using an additional RACF service machine.
- Auditing

  If you audit a large number of security relevant events, an additional RACF service machine can help improve the distribution of the overhead in recording the audit records. Each service machine has its own SMF minidisk.

IBM recommends that you install the RACF product and determine the overall system performance characteristics before you install multiple RACF service machines.

If you plan to install multiple RACF service machines, you should make changes to the RACSERV macro when you install RACF. This procedure is described in *RACF Program Directory*.

For information on setting up multiple RACF service machines, see Appendix A, "Setting Up Multiple RACF Service Machines," on page 151.

Dedicating RACF Service Machines for RACROUTE Request Processing

You may decide to dedicate a RACF service machine to handle only RACROUTE request processing from a service machine for a variety of reasons, including:

- The service machine issuing the requests is performance-sensitive. By dedicating a RACF service machine to it, processing time can be reduced for the RACROUTE issuer.
- The service machine issues many RACROUTE requests and you do not wish your general user population to be affected by possibly slower RACF services.

You can dedicate one RACF service machine to one RACROUTE issuer (for example, a VTAM® service machine or an SFS file pool server) or you can dedicate a RACF service machine to any subset of RACROUTE issuers.

This is optional; you are not required to dedicate RACF service machines for RACROUTE processing.

You can only dedicate a RACF service machine to process RACROUTE requests coded with the RELEASE= keyword specifying 1.9 or a later release. All RACROUTE requests with the RELEASE= 1.8.2 keyword are automatically processed by the RACF service machine that has been assigned to the RACROUTE issuer at LOGON.

To dedicate a RACF service machine for RACROUTE request processing:

1. Code CPUSE=NO on the RACSERV macro for that RACF service machine.

   Refer to *z/VM: RACF Security Server Macros and Interfaces* for more information on the RACSERV macro.
2. Copy the RACF SERVMACH file from the CMS Y-disk to the 191 disk of the RACROUTE issuer (or to another disk accessible to the RACROUTE issuer).
3. Change the user ID in the RACF SERVMACH file to the user ID of the dedicated RACF service machine.

Coordination of Commands

If you issue the SETROPTS command with any operand that changes the RACF database or issue the SETROPTS REFRESH command, the command is automatically propagated to all RACF servers that run on the same z/VM system, and to other systems in the same SSI cluster as the issuing system. Only the SETROPTS LIST command is not propagated.

On a system outside an SSI cluster, the action is not propagated to other systems that share the RACF database. You must issue the SETROPTS command separately for each system or restart the RACF servers on the other system or IPL the other system.

RACF automatically propagates the RVARY command to all RACF servers that run on the same z/VM system, and to all RACF servers that run on other systems in the same SSI cluster as the issuing system. RACF does not automatically propagate the RVARY command to z/VM systems that share the RACF database outside of an SSI cluster. You must issue the RVARY command separately for each system or restart the RACF servers on the other system or IPL the other system.

The RAC EXEC

z/VM users can enter RACF commands using the RAC EXEC (also called the RAC command processor).

The use of the RAC command can be controlled by creating a profile named RAC in the VMCMD class. For information on protecting RAC, see *z/VM: RACF Security Server Security Administrator's Guide*. When you enter a RACF command using RAC, the command is processed in the user's machine and also in the RACF service machine.

Processing in the end user's virtual machine allows users to:

- Capture command output
- Change the names and syntax of RACF commands.

Processing in the RACF service machine requires the RCMDS load module and allows system programmers to:

- Restrict, on an individual basis, users who can issue RACF commands
- Issue installation-written RACF commands through the RAC command.

Migration Note

Installations may have been using the RACFISPF module to establish an environment in which both RACF commands and CMS commands can be issued. Customer applications that use RACFISPF should migrate to the RAC EXEC.

The recommended way for general users to enter RACF commands on z/VM is to use RAC for line-mode command entry. This information describes the operation of RAC in the user's virtual machine.

"Using RAC in the RACF Service Machine" on page 61 provides information on RAC operating in the RACF service machine and ways for the system programmer to modify it.

Using RAC in the User's Virtual Machine

The RPIRAC module sends commands from the user's virtual machine to the RACF service machine for processing. Command output is then sent back to the user.

When the RACF product tape is installed on a z/VM system, the following files are placed on the system Y-disk

- RPIRAC MODULE
- RCMDRFMT EXEC
- RACOUTP EXEC
- RAC EXEC
- ICHSFSDF EXEC

These components make up RAC in the user's virtual machine. Preceding a RACF command with RAC allows users to issue commands without entering a RACF command session or using ISPF panels.

RAC captures the command output. Translation of the command output is possible with user modification of the RACOUTP EXEC.

How RAC Works

Any noninteractive RACF command can be used with RAC. Precede the RACF command with the word RAC. For example:

```
RAC SETROPTS LIST
RAC LU
```

RAC cannot be issued from the RACF service machine.

If you precede a command with RAC, the command output is presented on your screen, and put in an output file called RACF DATA. Your A-disk is the default file mode and it must be READ/WRITE. You should avoid sharing this minidisk with another user or unpredictable results may occur.

If you issue another command using RAC, the output is presented on your screen and is again available in RACF DATA. The first command's output (in RACF DATA) is overwritten (overwriting RACF DATA is also a default).

The RACOUTP EXEC, as provided by IBM, reads all the lines placed in the RACF DATA file, and displays it at the user's console.

Restrictions

- You cannot use RAC to switch SMF minidisks.
- You cannot use RAC to enter the BLKUPD command.
- The maximum length of a command you can issue from RAC is 3500 characters.
- The RAC command is an application that uses the SMSG communication protocol. If RAC is used within an application, that application should not use SMSG. All SMSGs received while the RAC command is processing a request to the RACF service machine are written to the RACF DATA file.

Modifying RAC in the User's Virtual Machine

There are global variables a user may change to tailor the RAC command processor to his user ID. For more information, see *z/VM: RACF Security Server Command Language Reference*

RAC and Its EXECs in the User's Virtual Machine

The product tape gives you the RCMDRFMT, RACOUTP, and ICHSFSDF EXECs. The installation exec places them on the system Y-disk.

- RACOUTP deals with command output.
- RCMDRFMT deals with command renaming and reformatting.
- ICHSFSDF deals with SFS commands.

Tailoring Command Output in the User's Virtual Machine

The RACOUTP EXEC always gets control after completion of a command that began with RAC. It is always called.

If output is not being appended to the RACF DATA file, RACOUTP lists the command output at the user's terminal. It reads the RACF DATA file into the program stack and displays each line with the REXX 'say' command. If the RACF DATA includes output from a propagated command then some additional filtering may occur. See "RAC (Enter RACF Commands on z/VM)" in *z/VM: RACF Security Server Command Language Reference* for more information. If output is being appended to the RACF DATA file, RACOUTP does not list the command output at the user's terminal.

Installations or users will be able to tailor their screen output and develop their own reports by modifying this EXEC.

Changing Command Names and Syntax in a User's Virtual Machine

The RCMDRFMT EXEC is a dummy EXEC that returns control to the RPIRAC module and sets a return code. The return code from the dummy EXEC is set to indicate no command translation has taken place.

To do translation or reformatting, write your own RCMDRFMT EXEC. You can use the sample EXEC called RCMDRFMT $EXEC$, by renaming it to RCMDRFMT EXEC.

How to Use RCMDRFMT

The RCMDRFMT $EXEC$ is provided on the product tape and contains sample code. A user can copy the EXEC from the system disk, rename it, and change the command syntax to suit.

The EXEC obtains access to the command processed by RAC, using the following REXX invocation as input:

```
PARSE ARG [input_line]
```

Note: The *input_line* value remains in mixed case if you entered it in mixed case.

RCMDRFMT must return with the syntax:

```
EXIT [return_code] [output_line]
```

Note:

1. The output line must have a 30-byte header of any character string appended to the command.
2. The command name must start in column 31, immediately following the 30-byte header.
3. The return code must be a 1-byte hexadecimal value and must contain one of the following values:

   ```
   X'00' or X'04'.
   ```

   Any other value causes the RCMDRFMT EXEC to fail the command.

   - If the return code is X'00', the command-translation EXEC directs the RAC EXEC to continue processing the command as it was entered by the user. The contents of the *output_line* variable are irrelevant.
   - If the return code is X'04', the command-translation EXEC has translated the command passed in *input_line* and reformatted it into a valid RACF command in *output_line.* Regardless of the contents of the *output_line*, it is treated and executed as a RACF command.

The RCMDRFMT EXEC provided by IBM provides *sample* code for translating the LIST, PERMIT and SEARCH commands and syntax.

For example:

A user enters:

```
RAC list user02 191 mini(acc
```

The following character string appears in the REXX variable *input_line* when the command-translation EXEC (RCMDRFMT) gets control:

```
(30-byte header) list user02 191 mini(acc
```

The command-translation EXEC passes back in *output_line*.

```
(30-byte header)  rlist vmmdisk user02.191 all
```

A user enters:

```
RAC give john read to user02 191 mini
```

and the command is converted to:

```
(30-byte header)   permit user02.191 class(vmmdisk) id(john) access(read)
```

Another example:

```
RAC find all tape
```

and the command is converted to:

```
(30-byte header)   search cla(tapevol)
```

Using RAC with SFS Files and Directories

When you use the RAC command processor with SFS files or directories, the ICHSFSDF EXEC is called. For a description of this EXEC, see “EXECs Supplied for Use on RACF for z/VM” on page 63.

Using RAC in the RACF Service Machine

This section provides information on the operation of the command processor operating in the RACF service machine and describes ways for the system programmer to exploit the new function.

When you install RACF, the load module RCMDS appears in the RACFLPA LOADLIB.

When the RPIRAC module passes commands to the RACF service machine, the RCMDS load module processes the requests and directs the output back to the user.

Installations may wish to place restrictions *on an individual basis* as to who can issue RACF commands. For example, all general users are allowed to issue the SEARCH command, but this command can be long-running and multiple requests may affect performance. Installations may wish to allow only users with the SPECIAL attribute the ability to issue the command. Installations may also choose to issue their installation-written command processors by using the RAC command. With RACF 1.9 or later, installations now have those abilities, but certain conditions must first be met.

Restricting Use of RACF Commands

To restrict the usage of RACF commands by users, the system programmer can write a command control exit called RPIRACEX. RPIRACEX is an exit point in the RACF service machine code. The text file for the exit must be link-edited into the RACFLPA LOADLIB as part of the RCMDS load module.

RPIRACEX Exit

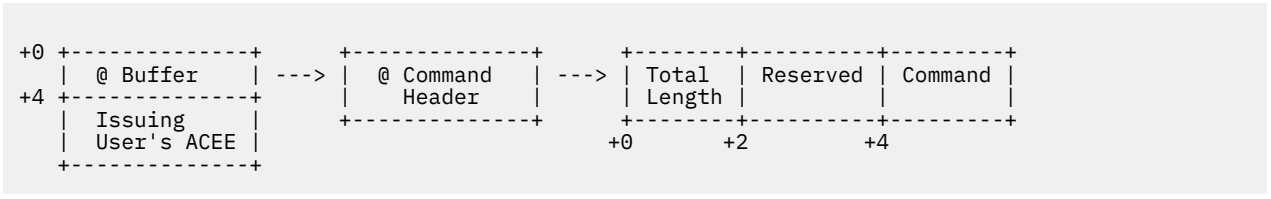
The command control exit, RPIRACEX, can fail a RACF command before RACF gets control, and can prevent the user from issuing the command.

The exit must be reentrant, have an AMODE of 24, and be capable of running in 370 mode. The exit must be link-edited into the RACFLPA load library. The exit is invoked in supervisor state, key 14.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see “Build or Link-Edit a Library” on page 166, using the following substitution values:

- For *fn* use **RPIRACEX**
- For *blist* use **RPIBLLPA**
- For *memname* use **RPIRACEX**

Upon entry to RPIRACEX, register 1 contains the address of a parameter list of pointers. The first full word of the parameter list is the address of a buffer. The buffer contains a pointer to the address of the command header. The second full word of the parameter list is the address of the issuing user's access-control environment element (ACEE).



The command header contains:

- Bytes 0 and 1—the total length, including 4 bytes for the header
- Bytes 2 and 3—reserved
- Bytes 4 to ?—the command string (including leading blanks).

Note: RPIRACEX must not modify the header or the command string. If it does, results are unpredictable.

On exit, register 15 tells RPIRCMDS whether to continue processing or to fail the command, as follows:

- If R15 = 0, continue processing; user authorized.
- If R15 ≠ 0, stop processing; user not authorized.

The RPIRACEX routine is called on every RAC invocation.

Adding Installation-Written Commands for the RAC Command Processor

If you are planning to create your own RACF commands, you must write a RACF command processor.

For use with RAC, this processor cannot use macros that generate terminal READs; for example, RDTERM or TGET macros. The command processor cannot prompt the user or the results are unpredictable.

To add a new command to RACFOBJ to be used with the RAC command processor, you need to perform local modifications to RPIRCMTB ASSEMBLE and RPIRCMTB TEXT. Follow the instructions in "Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List" on page 161, using the following substitution values. This process builds the RACFOBJ TXTLIB, RACFCMDS LOADLIB, and RACFLPA LOADLIB and copies the new ASSEMBLE file to the test build disk. You also need to add your new command to the RACFOBJ TXTLIB and RACFCMDS LOADLIB.

- For *fn* use **RPIRCMTB**
- For *blist* use **RPIBLOBJ** and **RPIBLCMD**
- For *memname* use **your new command name**
- For *nnnn* use **0002**

The RACF Command Session

z/VM users can enter RACF commands using the RACF command session.

This can be controlled by creating a profile named RACF in the VMCMD class. For information on protecting the RACF command session, see *z/VM: RACF Security Server Security Administrator's Guide*. When you enter a RACF command using the RACF command session, the command is processed in the user's machine and also in the RACF service machine.

Adding Installation-Written Commands for the RACF Command Session

If you are planning to create your own RACF commands, you must write a RACF command processor.

For use with RACF, this processor can use macros that generate terminal READS, which allow prompts to the user.

To add a new command to RACFOBJ to be used in the RACF command session, you need to perform local modifications to RPITMPTB ASSEMBLE and RPITMPTB TEXT. Follow the instructions in "Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List" on page 161, using the following substitution values. This process builds the RACFINTF LOADLIB and RACFCMDS LOADLIB and copies the new ASSEMBLE file to the test build disk. You also need to add your new command to the RACFOBJ TXTLIB and RACFCMDS LOADLIB.

- For *fn* use **RPITMPTB**
- For *blist* use **RPIBLOBJ** and **RPIBLCMD**

  Note: You need to use these to add your new command to these build lists.

  1. Perform the VMFBLD for RPIBLOBJ.
  2. Perform the VMFBLD for RPIBCMD.
- For *memname* use **your new command name**
- For *nnnn* use **0002**

EXECs Supplied for Use on RACF for z/VM

RACF for z/VM provides various EXECs on the product tape to facilitate installation, for use by the system administrators, auditors and general users, and for product maintenance.

To use an EXEC, type in the name of the EXEC and press the Enter key.

Auditing EXECs

EXEC
Description

RACDSMON
Executes the data security monitor (DSMON) program in the VM environment. See *z/VM: RACF Security Server Auditor's Guide* for details of how to run this EXEC.

RACFADU
Executes the RACF SMF data unload utility. See *z/VM: RACF Security Server Auditor's Guide* for more information.

RACRPORT
Executes the RACF report writer. See *z/VM: RACF Security Server Auditor's Guide* for more information.

SMFPROF
Profile EXEC for the RACFSMF virtual machine which does the SMF switching and archiving tasks. See *z/VM: RACF Security Server Auditor's Guide* for more information.

General Use EXECs

EXEC
Description

ICHDIRMV
Allows you to rename and relocate directories, subdirectories, and underlying files in the SFS environment. You can use this EXEC in place of the CMS RENAME DIRECTORY and CMS RELOCATE DIRECTORY commands when those commands are restricted. See *z/VM: RACF Security Server Security Administrator's Guide*.

ICHSFSDF
Works in conjunction with the RAC EXEC and is called when a RACF SFS command is issued. If the file pool ID or user ID has been omitted from an SFS format name, ICHSFSDF inserts the current file pool ID or the file space into the SFS format name.

You should not modify the ICHSFSDF EXEC.

ISPF
Allows users to start an ISPF panel session if ISPF is installed. The ISPF exec shipped is a sample (ISPFRACF EXECSAMP) that customers may tailor at installation time. See *z/VM: RACF Program Directory* for more information.

RAC
Allows users to enter RACF commands without entering a RACF command session. See *z/VM: RACF Security Server Command Language Reference*.

RACFLIST
Lists profiles for resources. See *z/VM: RACF Security Server General User's Guide* for more information.

RACFPERM
Grants or revokes authority to access resources. See *z/VM: RACF Security Server General User's Guide* for more information.

RACGROUP
Determines the ACIGROUP, if any, a user belongs to. See *z/VM: RACF Security Server General User's Guide* for more information.

RACOUTP
Displays the output of a RACF command that was issued with the RAC EXEC. It can also be used to tailor the output. The output is obtained from the RACF DATA file.

RACSEC
Displays the currently active security label for a user ID. For more information, see *z/VM: RACF Security Server Security Administrator's Guide*.

RCMDRFMT
Tailors the syntax of RACF commands if they are entered with the RAC EXEC. For more information, see the discussion earlier in this chapter.

Installation EXECs

The following EXECs are used during installation of RACF.

EXEC
Description

GENNUC
Generates the modified CMS which gets IPLed from the 490 disk of the RACF service machine.

RACALLOC
Allocates space on an OS-formatted minidisk for use as a RACF database.

RACDSF
OS-formats a minidisk for use as a RACF database.

RACINITD
Initializes an OS-formatted minidisk or a CMS-formatted FBA disk for use as a RACF database.

RACONFIG
Defines the configuration of the primary and backup RACF databases.

RACSETUP
Contains FILEDEFs and ACCESS commands for all the RACF databases.

RPIBLDDS
Builds or adds to a RACF database by executing the RACF statements generated by RPIDIRCT.

RPIDIRCT
Scans the CP directory and produces RACF statements to build the RACF database.

See *z/VM: RACF Security Server Security Administrator's Guide* for examples and more information.

For details on running RPIDIRCT to define OpenExtensions information, see *z/VM: RACF Security Server Security Administrator's Guide*.

Security Administration EXECs

EXEC
Description

ICHSFS
Migrates SFS installations to RACF. See *z/VM: RACF Security Server Security Administrator's Guide*.

RACFDBU
Used to unload the RACF database to a sequential file. For a description of how to use RACFDBU, see *z/VM: RACF Security Server Security Administrator's Guide*.

RACFDEL
Uses the output from IRRUT100 to generate commands designed to remove occurrences of a user ID from the RACF database. For a description of how to use RACFDEL, see *z/VM: RACF Security Server Security Administrator's Guide*.

RPIDELU
Used to execute the commands generated by RACFDEL. For a description of how to use RPIDELU, see *z/VM: RACF Security Server Security Administrator's Guide*.

System Programming EXECs

EXEC
Description

RACIPLXI
Automatically logs on the AUTOLOG2 virtual machine following RACF initialization. RACF only invokes this EXEC during initialization. It can be modified to issue messages appropriate to your installation.

RACFCONV
Updates the database templates. Executes the RACF utility IRRMIN00. See the discussion in the utilities chapters for more information on running this EXEC.

RACFSVRS
Used to initialize multiple RACF service machines.

RACSTART
Starts RACF. Issued from the RACF service machine.

RACSVRXI
Exercises control following an IUCV SEVER from CP on the CP to RACF path. The RACF service machine is the only machine that invokes this EXEC, but it can be modified to issue messages appropriate to your installation.

RACUT100
Lists all occurrences of a user ID or group name in a RACF database. Executes the RACF utility IRRUT100. See the discussion in the utilities chapters for more information on running this EXEC.

RACUT200
Copies a RACF database and identifies inconsistencies in the database. Executes the RACF utility IRRUT200. See the discussion in the utilities chapters for more information on running this EXEC.

RACUT400
Splits, merges, and copies a RACF database. Executes the RACF utility IRRUT400. See the discussion in the utilities chapters for more information on running this EXEC.

Using ACIGROUP

A z/VM installation may have ACIGROUP control statements in its users' directory entries.

ACIGROUP affects a user's LOGON, MDISK, and READER statements. ACIGROUP is *not* recommended, because the dual-registration panels do not accept ACIGROUP. For information on using RACROUTE with ACIGROUP, see *z/VM: Security Server RACROUTE Macro Reference*.

ACIGROUP and Installing RACF

During RACF initialization, the RPIDIRCT EXEC, which reads the CP source directory, is called. It produces an output file, RPIDIRCT SYSUT1, which consists of RACF commands that are used to build the RACF database. Any ACIGROUP entries are handled as follows:

- The ADDUSER default group is the ACIGROUP name:

```
ADDUSER SUE DFLTGRP(acigroup)
```

  rather than

```
ADDUSER SUE DFLTGRP(SYS1)
```

- All MDISK statements create profile names containing the ACIGROUP name:

```
RDEF VMMDISK acigroup.userid.vaddr
```

  rather than

```
RDEF VMMDISK userid.vaddr
```

- All READER statements are similarly constructed:

```
RDEF VMRDR acigroup.userid
```

Adding an ACIGROUP after RACF Is Installed

If your installation decides to add an ACIGROUP to a CP directory entry, it must also:

- Define the group to RACF (using ADDGROUP)
- Add new user IDs to the group (using ADDUSER)
- Connect previously defined user IDs to the group (using CONNECT).

Chapter 5. Utilities for the RACF Database

Attention:
Information on the block update command (BLKUPD) is found in *z/VM: RACF Security Server Diagnosis Guide*.
Information on the database unload utility (IRRDBU00) is found in *z/VM: RACF Security Server Macros and Interfaces* and *z/VM: RACF Security Server Security Administrator's Guide*.
Information on the data security monitor (DSMON) and the RACF report writer (RACFRW) is found in *z/VM: RACF Security Server Auditor's Guide*.
Information on the SMF data unload utility is found in *z/VM: RACF Security Server Auditor's Guide* and *z/VM: RACF Security Server Macros and Interfaces*.

The RACF utilities are used for maintaining, modifying, unloading, and monitoring the RACF database.

- **RACF database unload utility program (IRRDBU00)** unloads the RACF database to a sequential file. For information on how to use IRRDBU00, see *z/VM: RACF Security Server Macros and Interfaces*and *z/VM: RACF Security Server Security Administrator's Guide*.
- **RACF database-initialization utility program (IRRMIN00)** formats a non-VSAM DASD data set or creates a CMS file on an FBA disk for use as a RACF database. Use this utility with PARM=NEW to initialize a new, empty database. Use this utility with PARM=UPDATE when you update a RACF database template.
- **RACF cross-reference utility program (IRRUT100)** lists all the occurrences of a user ID or group name in the RACF database.
- **RACF database-verification utility program (IRRUT200)** identifies inconsistencies in the internal organization of a RACF database and provides information about the size and organization of a RACF database. You also can use this utility to create an *exact* copy of a RACF database, one that is no larger and no smaller and residing on the same device type.

  To make a larger or smaller copy of your database or to copy a database to a different device type, use IRRUT400.
- **Block-update utility command (BLKUPD)** modifies the records in a RACF database. You execute BLKUPD in a RACF command session. BLKUPD may be used to correct any inconsistencies that IRRUT200 finds in the RACF database. For information on how to use BLKUPD, see *z/VM: RACF Security Server Diagnosis Guide*.
- **Split/merge/extend utility program (IRRUT400)** copies a RACF database to a larger or smaller database, redistributes data from RACF databases, identifies inconsistencies in RACF databases, and physically reorganizes the database by bringing all the segments of a specified profile together. Use this utility to copy a database to a different device type.

  RACF allows up to four primary databases and up to four corresponding backup databases.

  Note: You should not split a RACF database residing on FBA DASD.

Note:

1. If you are sharing a database, the templates must match the latest level of code. The IRRMIN00 utility (for the latest release) updates the database templates. The templates are downwardly compatible. For example, if RACF 1.10 and RACF 5.3 are sharing a database, the templates must be at the 5.3 level, but your 1.10 system can successfully use the database.

© Copyright IBM Corp. 1993, 2025

Format minidisk utility (RACDSF)

This utility OS-formats a minidisk for use as a RACF database. While running RACF, to OS-format a disk you must:

1. Link to the RACF server 305 disk
2. Access the 305 disk
3. Have write access to the disk being formatted
4. Enter RACDSF

   The following screen appears:

```
                                RACF for VM - OS Formatting Utility          Screen 1 of 1

To OS format minidisks, type the virtual addresses and labels and press PF2.
Default labels will be used for labels not specified.

                     VADDR  LABEL
  Primary disks      ____ / ______
                     ____ / ______
                     ____ / ______
                     ____ / ______
  Backup disks       ____ / ______
                     ____ / ______
                     ____ / ______
                     ____ / ______
  Work disks         ____ / ______
                     ____ / ______
                     ____ / ______
                     ____ / ______


 PF1=Help      PF2=OS format  PF3=Exit
Enter CP/CMS Commands below:
====>
```

Figure 6. RACDSF OS-Formatting Utility

Fill in the device addresses and optional label fields (the label should conform to your installation's naming conventions). When you complete this screen, press **PF2** to OS-format the disks.

Note: The RACDSF utility should not be used for minidisks that reside on FBA (SCSI) DASD.

Allocate RACF dataset utility (RACALLOC)

This utility allocates the dataset on an OS-formatted minidisk for use as a RACF database. When running RACALLOC, to allocate a RACF database you must:

1. Link to the RACF server 305 disk
2. Access the 305 disk
3. Have access to the RACF server 200 disk
4. Have write access to the disk where the database is being allocated
5. Enter RACALLOC

   The following screen appears:

```
Is allocation for RACF primary , backup , or working data set?

Enter primary - to allocate a single primary RACF data set

Enter backup  - to allocate a single backup RACF data set

Enter Cuu     - to allocate a specific device address
                ( valid addresses are  200 211 212 213 300
                                       311 312 313 400 411
                                       412 413 )

Enter QUIT    - to terminate processing
```

Figure 7. RACACCOC Allocate Dataset Utility

Specify either primary, backup, or the device address and press **Enter** to have the RACF database allocated on the specified device.

Note: The RACALLOC utility should not be used for minidisks that reside on FBA (SCSI) DASD.

RACF Security Level Display Program (RACSEC)

The RACSEC utility can be used to display the current security label for the target virtual machine. If no userid is specified, the current virtual machine will be assumed as input.

Syntax

The syntax of RACSEC is as follows:

```
RACSEC {*|userid}
```

*****
Specifies that you want the current security label for your user ID.

userid
Specifies a user ID to obtain the current security label information for that user ID.

If you do not have a SECLABEL defined to your user ID, the following message is displayed:

```
RACSEC002I Userid userid is not currently logged on, or does not have a security label.
```

If a SECLABEL is defined to your user ID, the following message is displayed:

```
RACSEC004I  The security label for user userid is _seclabel_.
```

To query the security label of a user other than yourself, you must meet one of the following requirements:

- If RACF protection for DIAGNOSE X'A0' subcode X'30' is in effect, you must have READ access to the DIAG0A0.QUERYSEC profile in the VMCMD class.
- If RACF protection is not in effect for DIAGNOSE X'A0' subcode X'30', you must have privilege class A or B.

For more information, see this topic in the *z/VM: RACF Security Server Security Administrator's Guide*:

- Protecting the DIAGNOSE X'A0' Subcodes

RACF Database-Initialization Utility Program (IRRMIN00)

The IRRMIN00 utility initializes a RACF database. It can be used in two ways:

- PARM=NEW initializes a new, empty, database.
- PARM=UPDATE updates an existing database with a new set of RACF templates.

Important:

1. PARM=NEW processing formats an empty database. If you run IRRMIN00 PARM=NEW against an existing RACF database, then all data is destroyed.
2. If you run IRRMIN00 PARM=UPDATE against an existing RACF database that is shared with z/OS, then the RACF database can no longer be used on z/OS.

If you have split your database, you must run IRRMIN00 against each database defined in your database name table. If you have a backup database, you must also run IRRMIN00 against each part of the backup database.

Running IRRMIN00 always places a new set of templates in the RACF database. In order for RACF to begin using the new set of templates, you will need to restart the RACF service machine.

If you have an earlier release of RACF installed, you should use the latest version of IRRMIN00 to create or update the RACF database.

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/VM: RACF Security Server Command Language Reference*.

Running IRRMIN00 When PARM=NEW Is Specified

The RACF database-initialization program (IRRMIN00) formats a non-VSAM DASD data set or creates a CMS file on an FBA disk for use as a RACF database. Use this utility with PARM=NEW to initialize a new, empty, database.

You must run IRRMIN00 during the initial installation of RACF to format the RACF database. After RACF is installed, you can run IRRMIN00 to format an alternate RACF database.

Note: The installation process uses IRRMIN00 to format primary and backup RACF databases.

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 block
The header block (inventory control block).

Templates
The user, group, data-set, and general template definitions, plus six 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, writes the segment table, and updates the pointers and counts in the ICB block to reflect the new templates. The index blocks and profiles are not altered.

If the database you are updating is the active RACF database, IRRMIN00 obtains an exclusive RESERVE (enqueue) on it. If any copies of the RACF database blocks are in main storage, they are rendered incorrect and not referenced. RACF restores the database blocks as they are needed.

Diagnostic Capability

IRRMIN00 does not provide RACF database diagnostic information. If you suspect a RACF database error, start your problem determination by running the IRRUT200 utility described in "RACF Database-Verification Utility Program (IRRUT200)" on page 76.

For more information on RACF database diagnosis and correction, see Chapter 7, "Recovery Procedures," on page 139 and *z/VM: RACF Security Server Diagnosis Guide*.

Input for IRRMIN00

On z/VM

You can run IRRMIN00 from either the RACFVM user ID when RACF is inactive or from RACMAINT while RACF stays active in the RACFVM user ID.

You can run IRRMIN00 in either of two ways:

1. To format a new database, run the RACINITD EXEC, which starts IRRMIN00. The RACINITD EXEC starts IRRMIN00 with PARM=NEW.
2. To update the RACF database templates, run the RACFCONV EXEC, which starts IRRMIN00. The RACFCONV EXEC starts IRRMIN00 with PARM=UPDATE. IRRMIN00 formats the database, adds or updates the templates as required by the new release, and leaves the old profiles intact.

   Before running RACFCONV, run RACUT200 to verify the RACF primary and backup databases. A code 0 is returned from 'IRRUT200' if the RACF databases have no errors. The RACUT200 output report is sent to your virtual printer. If you received a non-zero return code, peek or XEDIT the output file, and do a quick locate for errors: search for "IRR62" messages and "LOCATIONS WITH POSSIBLE CONFLICTS". You might find errors in RACF PROFILEs, or RBA errors. These errors must be corrected before running the RACFCONV utility.

   Note: If there is any other activity on the RACF database while RACFCONV is running, the database might become corrupted. Take the following precautions to prevent corruption:

   a. If you update the templates on a standalone system, deactivate the database or FORCE the RACFVM userid before you run RACFCONV. Run RACFCONV from the RACMAINT userid.

   b. If you update the templates on a database that is shared with other systems or members of an SSI, FORCE RACFVM on all the systems that share the RACF database and then run RACFCONV from the RACMAINT userid.

   Note: RACF service machines load a copy of the RACF templates into virtual memory during RACF initialization. These copies of the RACF templates in virtual memory are not refreshed by running the RACFCONV utility. After you upgrade the RACF database template by running RACFCONV from the RACMAINT user, you must refresh the templates in virtual memory in the following way:

   - Restart all RACFVM users that share that RACF database (local, remote, and inside an SSI cluster) with the FORCE/XAUTOLOG commands.

IRRMIN00 requires no input. RACINITD and RACFCONV, which you use to start IRRMIN00, contain FILEDEFS for the files that contain the database templates. The RACINITD and RACFCONV utilities require the user specification of primary, backup, or a specific data-set virtual address as input. The RACF database to be formatted as the primary or backup database is defined in the RACONFIG EXEC. If you reformat an existing database, the database is updated in place.

Output from IRRMIN00

RACF writes the input images from the template definitions to the printer along with messages indicating errors or success.

The IRRMIN00 program sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.
4	(4)	The RACF database is usable. The attempted update to the template definitions would either produce the same result or downlevel the database.
C	(12)	The program encountered a terminating error. The RACF database was not formatted.
10	(16)	The output database could not be opened. The RACF database was not formatted.
14	(20)	The RACF database is usable. The attempted update to the template definitions is not supported. See the "MIN00U OUTPUT" file for details.

Note: RACFCONV returns RC 0 if IRRMIN00 returned RC 0 or RC 4, and displays the state of the database in a message. In all other cases, the RC of IRRMIN00 is returned.

RACF Cross-Reference Utility Program (IRRUT100)

IRRUT100 is a RACF utility program that lists all occurrences of a user ID or group name that are in a RACF database. It uses the RACF manager to access the RACF database and locate possible occurrences of a user ID or group name.

An alternative to using IRRUT100 is to use the database unload utility, IRRDBU00. It provides a sequential file of the database that an installation can manipulate to obtain additional and more complex reports. For more information on IRRDBU00, see *z/VM: RACF Security Server Macros and Interfaces* and *z/VM: RACF Security Server Security Administrator's Guide*.

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.

As IRRUT100 executes, it issues one Reserve per profile, not a continuous Reserve. When IRRUT100 has finished copying a profile, it releases the Reserve on it. Thus, the database *is* accessible, depending on the performance options set at your installation and other ongoing system activity.

IRRUT100 produces a cross-reference report that describes the occurrences of each user ID or group name specified. The letter G in parentheses follows each generic profile name.

For groups, IRRUT100 provides information on the following occurrences:

- The group name exists in the RACF database.
- The group is a subgroup of group *xx*.
- The group is a superior group of group *xx*.
- The group is the default group for user *xx*.
- The group is a connect group for user *xx*.
- The group was the connect group when the user created data set profile *xx*.
- The group name is the high-level qualifier of data set profile *xx*.
- The group has standard access to data set profile *xx*.
- The group has standard access to general resource *xx*.
- The group is the owner of user *xx*.

- The group is the owner of group *xx*.
- The group is the owner of data set profile *xx*.
- The group is the owner of general resource *xx*.
- The group is the owner of connect profile *xx*.
- The group exists in the conditional access list of general resource profile *xx*.
- The group exists in the conditional access list for data set profile *xx*.
- The group is the resource owner of profile *xx*.

For user IDs, IRRUT100 provides information on the following occurrences:

- The user ID exists in the RACF database.
- The user is the owner of group *xx*.
- The user is a member of (connected to) group *xx*.
- 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 ID is the second qualifier of FILE profile *xx*.
- The user ID is the second qualifier of DIRECTRY profile *xx*.

See Figure 8 on page 76 for a sample output of the printed report that IRRUT100 produces.

Exit Routine

RACF provides a preprocessing exit for an installation-written routine when the IRRUT100 utility is invoked. For more information, see “Command Preprocessing Routine ICHCNX00” in Chapter 6, “RACF Installation Exits,” on page 107.

Diagnostic Capability

IRRUT100 does not provide RACF database diagnostic information. It can read many of the profiles in the database and may identify profiles with errors. If you suspect a RACF database error, start your problem determination by running the IRRUT200 utility described in “RACF Database-Verification Utility Program (IRRUT200)” on page 76.

For more information on RACF database diagnosis and correction, see Chapter 7, “Recovery Procedures,” on page 139 and *z/VM: RACF Security Server Diagnosis Guide*.

The Work CMS File

Records in the work file 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'
Reserved

X'18'
Qualifier of the general resource profile (This is used only for FILE and DIRECTRY profiles)

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

To use IRRUT100, execute the RACUT100 EXEC from a user ID that has links to RACFVM's 305 and 490 disks and to the RACF database disks (RACFVM's 200, 300, and any other database disks). The user ID must also have:

- The 490 disk IPLed

Do not issue RACUT100 from the RACMAINT user ID because the result may not be accurate. RACUT100 prompts you for the virtual address or addresses of the RACF database.

Enter the addresses of all primary and backup RACF databases in response to prompts. After you have entered all the addresses, RACUT100 places you in XEDIT mode to allow you to create the input for the IRRUT100 SYSIN file as shown below. RACUT100 DDR-copies the RACF database to a TDISK, and uses the copy to produce IRRUT100 reports. The output will be a print file.

The format of IRRUT100 SYSIN file 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 terminates the utility program.

IRRUT100 Example

In this example, IRRUT100, running under RACUT100, locates all occurrences of the group name RACG0001 and the user ID RACU0002 in the RACF database and prints these occurrences on the system output device.

1. Enter: racut100.
2. RACUT100 asks you for the INPUT RACF dataset device address.
3. Enter the virtual address; for example, 200.
4. RACUT100 asks you for the next INPUT RACF dataset device address.
5. Enter a virtual address; for example, 300.
6. RACUT100 asks you for the next INPUT RACF dataset device address.

7. Enter: end.
8. RACUT100 puts you in XEDIT to create, or add to IRRUT100 SYSIN.
9. Add the following two lines to the file:

   racg0001 racu0002
   /end
10. File IRRUT100 SYSIN.

Ignore messages RPISMF050E and RPISMF054I.

```
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 dataset 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 dataset profile USERMID.GROUP.TEST (G)
 In notify field of dataset profile HILDE.NOTIFY.CNTL
 In conditional access list of dataset 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
 Qualifier of general resource profile FILE  FP1.USER1.DIR1.SIVLE.MEM
 Qualifier of general resource profile DIRECTRY FP1.USER1.**  (G)
 Owner of user USER2
 User entry exists

 (G) - Entity name is generic.
1
```

Figure 8. Sample Output from IRRUT100

RACF Database-Verification Utility Program (IRRUT200)

IRRUT200 is a RACF utility program that you can use to identify inconsistencies in the internal organization of a RACF database and also to make an exact copy of the RACF database. To examine your database, you must use the release level of IRRUT200 that corresponds to the release level of your database. It performs the following functions:

- Validates and reports errors found in the relative byte addresses (RBAs) of each segment of all profiles

- Validates that index entries point to the correct profile
- Validates the database format
- Issues return codes to signal validation errors
- Scans the index blocks and prints information about problems with the index-block chains
- Compares the segments of the database that are actually in use to the segments allocated according to the BAM blocks, and prints information about inconsistencies
- Creates a backup copy of a RACF database
- Creates an enhanced, formatted index report displaying the 255-byte profile name and profile type information.

Copying a RACF Database

IRRUT200 creates an exact block-by-block copy of the RACF database. This exact copy can help performance when you are maintaining statistics on your backup database.

Note:

1. IRRUT200 will serialize on the RACF database if it has been set up for sharing. See "Database Considerations" on page 41 for information on how to correctly set up your RACF database for sharing. **Note also that a RACF database residing on FBA (SCSI) DASD cannot be shared.**
2. If the RACF database has been set up for sharing then IRRUT200 can be used when there is update activity on the database being copied. However to minimize disruption it is recommended that IRRUT200 only be used during times of low database activity.
3. If the RACF database has NOT been set up for sharing then IRRUT200 will NOT serialize on the RACF database and so should NOT be used when there is update activity on the database being copied. To maintain data integrity of the target database in this situation, IRRUT200 copy should only be run from the RACFVM server virtual machine with RACF deactivated.

IRRUT200 can be used only if you are creating a copy of the database that is the same size and on the same device type as the input database. 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 database size or to copy a database to a different device type, use IRRUT400.

The target of the copy should not be an active RACF database. If you need to refresh an active RACF database, issue an RVARY INACTIVE before running IRRUT200. After utility processing completes, issue RVARY ACTIVE.

Diagnostic Capability

IRRUT200 is designed to detect errors in the internal organization of the RACF database. If you suspect a RACF database error, start your problem determination by running this utility. When the job completes, inspect the utility return code. If a zero return code is returned, it is likely that your database is fine. If a non-zero return code is returned, review the output produced by the utility. Searching for "IRR62" messages can bring you quickly to the reported error.

See Chapter 7, "Recovery Procedures," on page 139 and *z/VM: RACF Security Server Diagnosis Guide* for more information on RACF database diagnosis and correction.

Additional diagnostic information:

- IRRUT200 checks most of the internal organization of the RACF database. However, it does not verify every field within a profile. Therefore it is possible for IRRUT200 to produce a zero return code even though the RACF database contains a profile in error.

  If you suspect your database contains such an error, run the RACF database unload utility, (IRRDBU00). The IRRDBU00 utility must read every profile in the database and can identify profiles with errors.

  For more information, see the IRRDBU00 description in *z/VM: RACF Security Server Security Administrator's Guide*.

- If IRRUT200 reports errors on upper level index blocks only, you can use the IRRUT400 utility to create a new copy of the RACF database.

Processing Considerations for Databases from Other Systems

For proper utility operation, the enhanced-generic-name (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 back-up) 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 database in the database set that contains the database that you are processing.
  – Read record zero by issuing the READ X'00' BLKUPD subcommand.
  – List the 194th 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 may cause a question mark (?) to be displayed as a part of a formatted index entry.

Using IRRUT200

If you are using IRRUT200 to verify or copy a shared database, execute RACUT200 from the RACMAINT user ID or from a user ID with links to RACFVM's 305 disk, 490 disk, and the RACF database disk you want to copy or verify.

To copy a non-shared database using IRRUT200, do the following:

- Logon to the RACFVM server.
- Deactivate RACF by IPLing RACFVM's 490 disk.
- Issue RACUT200 from RACFVM.
- After RACUT200 has executed, IPL RACFVM's 490-disk.
- Reactivate RACF by issuing RACSTART EXEC from RACFVM.

As it executes, RACUT200 prompts you for the following:

- The option to verify or copy the database
- The virtual address of the preallocated minidisk address that contains the database you want to copy or verify
- If you are copying, the minidisk onto which you want the database to be copied.

Input and Output

IRRUT200 uses the following input:

- A control file, called RACVERFY FILE, which contains the utility control statements that indicate the functions to be performed. If you are verifying a RACF database, the control statements are contained in the RACVERFY FILE. As RACUT200 executes, it gives you an opportunity to create a RACVERFY FILE, or to use an existing RACVERFY FILE.
- A RACF database. (The RACONFIG EXEC defines the RACF database. See *RACF Program Directory* for a description of how RACONFIG defines them.)

IRRUT200 produces the following output:

- A diagnostic error message displayed at your terminal.
- A print file for printing statistical data and the results of the IRRUT200 operations.

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. These statements are contained in the RACVERFY FILE.

Utility Control Statement for IRRUT200

```
INDEX [FORMAT]
I
```

where:

INDEX specifies you want the index scan function performed.
FORMAT specifies a formatted listing of all the index blocks.
Only one blank can separate INDEX and FORMAT.
You can use only columns 1 through 72.

Utility Control Statement for IRRUT200

```
MAP [ALL]
M
```

where:

MAP specifies you want the BAM/allocation verification performed.
ALL specifies that you want the encoded map for each BAM block in the RACF data set printed.
Only one blank can separate MAP and ALL.
You can use only columns 1 through 72.

Utility Control Statement for IRRUT200

```
END
```

where:

END terminates the utility program.

You can use only columns 1 through 72.

Scanning the Index Blocks

When an index block scan is requested, IRRUT200 verifies that the following are all true:

- The pointer to every index block is a multiple of 4096.
- Every index block begins with the value X'8A'.
- Every index entry name has a valid length.
- Every pointer entry in the index block is preceded by the value X'6'*x* (*x* may 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 index entry must have a nonzero segment count.

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/VM: RACF Security Server 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.

Additionally, summary statistics about the index are provided. These statistics may not be representative of the entire database 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 a RACF database. 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. This information is provided for each index entry within the block:

- The offset of the entry within the block.
- The front-end compression count.
- The entry name (with generic entry names followed by a G in parentheses).
- 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 9 on page 81 for sample output that IRRUT200 produces when you request formatted index blocks.

```
                                        ****  INDEX BLOCK VERIFICATION  ****
-                                    ****  SCAN OF INDEX BLOCKS AT LEVEL 01  ****

 BLOCK WITH RBA OF 00000000E000

 OFFSET  COMP.                          ENTRY NAME                                      RBA           BAM
         COUNT                                                                                  BLOCK  BYTE BIT
  00E    000  ADAM                                                                  00000000D000   00   02E   0
  026    000  ADRIENNE                                                              00000000DC00   00   02F   4
  040    000  ALAN                                                                  00000000DB00   00   02F   3
  058    000  ALEX                                                                  00000000DA00   00   02F   2
  070    000  BRUCE                                                                 00000000DE00   00   02F   6
  08A    000  CHRIS                                                                 00000000DD00   00   02F   5
  0A1    000  CHUCK                                                                 00000000D900   00   02F   1
  0BC    000  DASDVOL -D001                                                         00000000D400   00   02E   4
  0D5    000  DASDVOL -D002                                                         00000000F000   00   032   0
  0ED    000  ERICA                                                                 00000000DF00   00   02F   7
  104    000  GENE                                                                  00000000F100   00   032   1
  11C    000  IBMUSER                                                               00000000F200   00   032   2
  136    000  JEFF                                                                  00000000F500   00   032   5
  157    000  JOE                                                                   00000000F600   00   032   6
  178    000  JOHN                                                                  00000000F700   00   032   7
  199    000  LAURIE                                                                00000000D700   00   02E   7
  1B4    000  SECLABEL-SYSHIGH                                                      00000000D100   00   02E   1
  1D8    000  SECLABEL-SYSLOW                                                       00000000D200   00   02E   2
  1FB    000  SIVLE                                                                 00000000D300   00   02E   3
  21F    000  SYSCTLG                                                               00000000D600   00   02E   6
  23A    000  SYS1                                                                  00000000F300   00   032   3
  252    000  VSAMDSET                                                              00000000D500   00   02E   5
  26E    000  255 X'FF'S
  37A         SEQUENCE SET POINTER                                                  000000000000

 TOTAL NAMES IN THIS BLOCK-023. UNUSED BYTES-3151. AVERAGE NAME LENGTH-018.
 LEVEL NUMBER-01. DISPLACEMENT TO LAST KEY-037A. DISPLACEMENT TO FREE SPACE-0383
 (G) - ENTITY NAME IS GENERIC

                                            ****  SEQUENCE SET RBAS  ****
       RBA  00000000E000
                                        ****  INDEX FUNCTION STATISTICS  ****

 TOTAL NUMBER OF NAMES IN RACF DATA SET 00000023
 TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000001
 AVERAGE NUMBER OF NAMES PER INDEX BLOCK 023
 AVERAGE NAME LENGTH 018
 AVERAGE NUMBER OF UNUSED BYTES PER INDEX BLOCK 3151
 TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000001
```

Figure 9. Sample Output of Formatted 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 will be checked between the Level 1 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 index entry must have a nonzero segment count.

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/VM: RACF Security Server 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 database segments and the status of the segments as defined by the masks in the BAM blocks. The codes 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.

.
The segment is defined as allocated by the BAM but is actually unallocated.

+
The segment is defined as unallocated by the BAM but is actually allocated.

I
Refers to an index block with the level in the next positions. This symbol implies an asterisk (*).

B
Refers to a BAM block. This symbol implies an asterisk (*).

Symbol
Meaning

T
Refers to a template block. This symbol implies an asterisk (*).

F
Refers to the first block (ICB). This symbol implies an asterisk (*).

S
Segment table

–
Refers to an index, BAM, or first block that is defined as unallocated but is actually allocated.

$
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 invalid, so its type is unknown.

%
Refers to block that is defined as unallocated but is actually allocated. The block is invalid, 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.

/
Undefined space.

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 summary statistics concerning the RACF database:

- 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 database.

- The total number of level one index blocks.
- The number of profiles of each type in the database.
- If the database contains profiles with an undefined class type (the class is not in the active class descriptor table), the class number rather than the class name is the identifier.
- The percentage of space used on a RACF database.

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 database be printed. If inconsistencies are found, a table of conflict messages follows.

See Figure 10 on page 83 for a sample printout of the encoded map that IRRUT200 produces with MAPALL specified.

```
                                        ****  BAM BLOCK VERIFICATION  ****
            ****  SYMBOL LEGEND  ****

          *   BAM=ALLOC   , ACTUAL=ALLOC
          0   BAM=UNALLOC , ACTUAL=UNALLOC
          .   BAM=ALLOC   , ACTUAL=UNALLOC
          +   BAM=UNALLOC , ACTUAL=ALLOC
          I   INDEX BLOCK WITH LEVEL IN NEXT POSITIONS
          B   BAM BLOCK
          T   TEMPLATE BLOCK
          S   SEGMENT TABLE BLOCK
          F   FIRST BLOCK (ICB)
          -   BAM=UNALLOC , ACTUAL=ALLOC  I,B,OR F BLK
          $   BAM=UNALLOC , ACTUAL=ALLOC  SPECIAL BLK
          ?   BAM=ALLOC   , ACTUAL=ALLOC  UNKNOWN BLK
          %   BAM=UNALLOC , ACTUAL=ALLOC  UNKNOWN BLK
          @   BAM=ALLOC   , DUPLICATE ALLOCATION
          #   BAM=UNALLOC , DUPLICATE ALLOCATION
          /   UNDEFINED STORAGE
-BLOCK 000 RBA 00000000C000
 014 FFFFFFFF FFFFFFFF TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT
 021 TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT SSSSSSSS SSSSSSSS BBBBBBBB BBBBBBBB
 02E ******** ******** I1111111 11111111 ******** 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 03B 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 048 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 055 00000000 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 00000000
 06F 00000000 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 00000000
 089 00000000 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 00000000
 0A3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0B0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0BD 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0CA 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0D7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0E4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0F1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0FE 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 10B 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 118 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 125 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 132 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 13F //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 14C //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 159 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 166 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 173 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
  .
  .
  .
 F91 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 F9E //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FAB //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FB8 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FC5 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FD2 //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FDF //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FEC //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// //////// ////////
 FF9 //////// //////// //////// //////// //////// //////// //////// ////////

                                        ****  MAP FUNCTION STATISTICS  ****
0NUMBER OF BAM BLOCKS DEFINED 001
 LAST BAM THAT DEFINES USED SPACE - RBA 00000000C000
 RACF DATA SET IS  11 PERCENT FULL.
 TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000001
 TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000001
 NUMBER OF GROUP    ENTRIES - 0000003
 NUMBER OF USER     ENTRIES - 0000013
 NUMBER OF DASDVOL  ENTRIES - 0000003
 NUMBER OF SECLABEL ENTRIES - 0000003
```

Figure 10. Sample Output of Encoded BAM Map Produced by IRRUT200

Examples of IRRUT200 usage

The following examples show common IRRUT200 scenarios.

Copying a non-shared RACF database

If the RACF database is not set up for sharing, IRRUT200 will not serialize on the RACF database. IRRUT200 should not be used when there is update activity on the database being copied. To maintain data integrity of the target database in this situation, IRRUT200 copy should only be run from the RACFVM server virtual machine with RACF deactivated. To deactivate the RACFVM server, ensure RACSTART is not running on RACFVM by logging on RACFVM and issuing "IPL 490" (see Step “1” on page 84) to stop RACSTART from running.

To copy the primary database (on the 200 disk) to the backup database (on the 300 disk) using IRRUT200, logon to the RACFVM server and perform the following steps:

1. Enter `#CP I 490`.
2. Issue the RACUT200 command to copy the 200 disk to the 300 disk.

   Non-interactive method for copying a database:
   You can use the following RACUT200 syntax to accomplish this non-interactively:

   ```
   RACUT200 COPY input_address output_address
   ```

   The parameters in this syntax have the following functions:

   COPY
   Copies the database from *input_address* to *output_address*.

   input_address
   Specifies the device address of the database to be copied.

   output_address
   Specifies the device address to which the database is to be copied.

   The following non-interactive example copies the database from input device address 200 to output device address 300:

   ```
   RACUT200 COPY 200 300
   ```

   Interactive method for copying a database:
   Alternatively, you can accomplish this interactively by entering the RACUT200 command without parameters and then responding to system prompts. Like the non-interactive example, the following interactive example copies the database from input device address 200 to output device address 300:

   a. Enter `RACUT200`.
   b. Reply `NO` to the 'Do you want to Verify a RACF database?' prompt.
   c. Reply `200` to the 'Enter the Input device address' prompt.
   d. Reply `300` to the 'Enter the Output device address' prompt.
   e. Reply `YES` to the 'Do you wish to continue?' prompt.

   Messages `RPIRND003E` and `IRR62009I` can be ignored.

   Return code from 'IRRUT200' = 0 should be issued if successful.
3. Enter `#CP I 490`.
4. Enter `RACSTART`.
5. Enter `#CP DISC`.

Copying a shared RACF database

When the RACF database has been set up for sharing, IRRUT200 will serialize on the RACF database. This means that IRRUT200 can be used when there is update activity on the database being copied and RACFVM is running as usual. However, to minimize disruption it is recommended that IRRUT200 only be used during times of low database activity.

To copy the primary database (on the 200 disk) to the backup database (on the 300 disk), logon to RACMAINT or any userid with the authority to link RACFVM's 305 disk, 490 disk, and the database disks. Enter the following to link and access the disks if not already available:

1. Enter `LINK RACFVM 490 490 RR`.
2. Enter `LINK RACFVM 305 305 RR`.
3. Enter `LINK RACFVM 200 200 RR`. Enter the read password if required.
4. Enter `LINK RACFVM 300 300 MW`. Enter the multi password if required.
5. Enter `I 490`.
6. Enter `ACCESS 305 B`.
7. Enter `ACCESS 190 T`.
8. Issue the RACUT200 command to copy the 200 disk to the 300 disk:

   Use either the non-interactive instructions or the interactive instructions in "Copying a non-shared RACF database" on page 84 to copy the database.

   Messages RPIRND003E and IRR62009I can be ignored.

   Return code from 'IRRUT200' = 0 should be issued if successful.

Making a Point-In-Time Backup Copy of the Primary RACF Database

You might want to make a "point-in-time" backup copy of the primary RACF database on a temporary disk before running RACFCONV so that this backup can be used for recovery in the case of a RACFCONV error.

Before you make a point-in-time backup, take into account whether you are using a shared or non-shared RACF database. Because RACUT200 behaves differently for a shared RACF database versus a non-shared RACF database, a slightly different set up is required for each case before creating the point-in-time backup copy.

If the RACF database you plan to copy (VDEV 200 in this example) is not shared, it can not be active during the copy. In this case, you should do the RACUT200 copy from the RACFVM user ID. Deactivate the RACFVM server by issuing the following commands:

1. `LOGON RACFVM`
2. `IPL 490`

If the RACF database you plan to copy (VDEV 200 in this example) is shared, RACFVM can be left up and running as usual while you copy the database. In this case, LOGON to RACMAINT or to any user ID with the authority to link to RACFVM's 305 disk, 490 disk, and the database disks. Enter the following commands to link and access the disks, if they are not already available:

1. `LINK RACFVM 490 490 RR`
2. `LINK RACFVM 305 305 RR`
3. `LINK RACFVM 200 200 RR`. Enter the read password, if required
4. `IPL 490`
5. `ACCESS 305 B`
6. `ACCESS 190 T`

At this point we are ready to create a point-in-time backup. In the following example we will make a point-in-time backup of the primary RACF database (VDEV 200) on a temporary disk (VDEV 400). These steps are issued on the RACMAINT user ID. The commands you issue appear in bold text, followed by the responses you see on your terminal screen.

1. Define a temporary disk of the same type as VDEV 200; in this example, this is a 17-cylinder MDISK on 3390 DASD as shown by the QUERY DISK command:

```
q disk
LABEL  VDEV M  STAT   CYL TYPE BLKSZ  FILES  BLKS USED-(%) BLKS LEFT
RACF   200  R   R/W    17 3390           OS
```

```
def t3390 400 17
DASD 0400 DEFINED
Ready;

format 400 e
DMSFOR603R FORMAT will erase all files on disk E(400). Do you wish to
continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
tmp400
DMSFOR733I Formatting disk E
DMSFOR732I 17 cylinders formatted on E(400)
Ready;
```

2. Format the temporary disk with RACDSF to get an OS formatted MDISK:

```
racdsf
```

The following screen is displayed:

```
          RACF for VM - OS Formatting Utility      Screen x

 To OS format minidisks, type the virtual addresses and labels and press PF2
 Default labels will be used for labels not specified.

                         VADDR  LABEL
         Primary disks   ____ / ______
                         ____ / ______
                         ____ / ______
         Backup disks    ____ / ______
                         ____ / ______
                         ____ / ______
                         ____ / ______
         Work disks      ____ / ______
                         ____ / ______
                         ____ / ______

 PF1=Help     PF2=OS format     PF3=Exit
```

In the Work Disks fields, specify "400" as the VADDR and specify "TMP400" as the LABEL. Then press PF2 and you will see the following:

```
          RACF for VM - OS Formatting Utility      Screen x

 To OS format minidisks, type the virtual addresses and labels and press PF2
 Default labels will be used for labels not specified.

                         VADDR  LABEL
         Primary disks   ____ / ______
                         ____ / ______
                         ____ / ______
         Backup disks    ____ / ______
                         ____ / ______
                         ____ / ______
                         ____ / ______
         Work disks      0400 / TMP400
                         ____ / ______
                         ____ / ______

 PIDSF004I All specified disks have been OS-formatted successfully.
 PF1=Help     PF2=OS format     PF3=Exit
```

3. Issue RACALLOC against the temporary disk:

```
racalloc

Is allocation for RACF primary , backup , or working data set?

Enter primary - to allocate a single primary RACF data set

Enter backup  - to allocate a single backup RACF data set
```

```
Enter Cuu     - to allocate a specific device address
              ( valid addresses are  200 211 212 213 300
                                     311 312 313 400 411
                                     412 413 )
Enter QUIT    - to terminate processing

400
Allocation for RACF data set  RACF.DATASET  ends
The return code is  0
Ready;
```

4. Issue the RACUT200 command to copy the 200 disk to the 400 disk, where the 400 disk becomes the point-in-time backup copy.

   Note: The number of cylinders specified for input and output disks must be the same. Disks created should not have any overlapping cylinder ranges to avoid conflicts.

   Non-interactive method for copying a database:
   You can use the following RACUT200 syntax to accomplish this non-interactively:

   ```
   RACUT200 COPY input_address output_address
   ```

   The parameters in this syntax have the following functions:

   COPY
   Copies the database from *input_address* to *output_address*.

   input_address
   Specifies the device address of the database to be copied.

   output_address
   Specifies the device address to which the database is to be copied.

   The following non-interactive example copies the database from input device address 200 to output device address 400:

   ```
   RACUT200 COPY 200 400
   ```

   Interactive method for copying a database:
   Alternatively, you can accomplish this interactively by entering the RACUT200 command without parameters and then responding to system prompts. Like the non-interactive example, the following interactive example copies the database from input device address 200 to output device address 400:

   ```
   racut200

   This exec is used to Verify and create a backup
   Copy of a RACF data base.

   Press Enter to continue....

   Do you want to Verify a RACF data base?

   Enter YES or NO or QUIT
   no

   Enter the  Input  device address
   200

   Enter the  Output  device address
       or
   Enter a Null line to Bypass Copy
   400

   RACF.DATASET  is the input Racf data set at virtual address '200'

   RACF.DATASET  is the output Racf data set at virtual address '400'

   Do you wish to continue?

   Enter YES or NO
   yes
   ```

```
Processing of  RACF.DATASET  begins
Program 'IRRUT200' is being executed - Please wait -

NOTE:  Message RPIRND003E for TTR conversion failure on input disk is
       expected

NOTE:  Message DMSLOS013E is expected when ICHRRCDE is not available
       on your system.

CSTSET001I CMS SUB-TASKING SUPERVISOR INITIALIZED.
CSTINT003I INITIATOR ACTIVATED.
RPIRND003E - TTR conversion failed for block number 00000BDC on disk
0200.
IRR62009I - EOF on SYSIN - processing terminated
CSTINT004I PROGRAM 'IRRUT200' ENDED. COMPLETION CODE = 000000.
CSTINT006I NO MORE SUB-TASKS.
CSTTER001I CST TERMINATED.

Processing of  RACF.DATASET  completes
Return code from 'IRRUT200' = 0
Ready;
```

Verifying a database with RACF active

Verifying a RACF database can be done from RACMAINT userid or any userid with the authority to link RACFVM's 305 disk, 490 disk, and the database disks. In this example, the RACF primary database (on the 200 disk) will be verified. Enter the following to link the disks if not already available:

1. Enter `LINK RACFVM 490 490 RR`.
2. Enter `LINK RACFVM 305 305 RR`.
3. Enter `LINK RACFVM 200 200 RR`. Enter the read password, if required.
4. Enter `IPL 490`.
5. Enter `ACCESS 305 B`.
6. Enter `ACCESS 190 T`.

Execute RACUT200 to verify the database.

Non-interactive method for verifying a database:

You can use the following RACUT200 syntax to accomplish this non-interactively:

```
RACUT200 VERIFY input_address
```

The parameters in this syntax have the following functions:

VERIFY
Verifies the database at *input_address*.

input_address
Specifies the device address of the RACF database to be verified.

This non-interactive example verifies the database at device address 200:

```
RACUT200 VERIFY 200
```

Interactive method for verifying a database:

Alternatively, you can accomplish this interactively by entering the RACUT200 command without parameters and then responding to system prompts. Like the non-interactive example, this interactive example verifies the database at device address 200:

1. Enter `RACUT200`.
2. Reply YES to the 'Do you want to Verify a RACF database?' prompt.
3. If a RACVERFY FILE input file exists, you will be given the option to reuse it or overlay it. If a RACVERFY FILE does not exist, one will be created and XEDIT will be entered. Type `FILE` when editing is complete.

4. Reply 200 to the 'Enter the Input device address' prompt.
5. Press Enter to bypass copy.
6. Reply YES to the 'Do you wish to continue?' prompt.

Messages RPIOPN003E and IRR62003I can be ignored. You may also get messages DMSLOS013E and IRR62064I.

Return code from 'IRRUT200' = 0 should be issued if successful.

The IRRUT200 output report will be sent to your virtual printer.

Note: It is recommended that verification is run at a time of low RACF activity, or against a database that is inactive to RACF.

Verifying a database with RACF inactive

To verify the primary database (on the 200 disk), logon to the RACFVM server and enter the following:

1. Enter `#CP I 490`.
2. Use either the non-interactive instructions or the interactive instructions in "Verifying a database with RACF active" on page 88 to verify the database, then proceed with the following step.
3. Enter `#CP I 490`.
4. Enter `RACSTART`.
5. Enter `#CP DISC`.

IRRUT200 Return Codes

The IRRUT200 program sets the following return codes:

Hex	(Decimal)	Meaning
0	(0)	Successful completion.
4	(4)	A validation error was discovered while processing the RACF data set.
8	(8)	A severe error occurred.
	(12)	An I/O error occurred.

Note: See *z/VM: RACF Security Server Messages and Codes* for the IRRUT200 messages.

The RACF Database Split/Merge/Extend Utility Program (IRRUT400)

IRRUT400 is a RACF utility program that supports multiple RACF databases.

It performs the following functions:

- Copies a RACF database to a larger or smaller database, provided there is enough space for the copy. See the restriction for z/VM FBA devices in the "Executing the Split/Merge/Extend Utility" on page 91.
- Redistributes data from RACF databases. For example, IRRUT400 can split a single RACF database into multiple RACF databases, merge multiple RACF databases into fewer RACF databases, or rearrange RACF profiles across the same number of input and output RACF databases. Though the utility allows a maximum of 255 input databases and 255 output databases, RACF allows up to four primary database and up to four corresponding backup databases.

  Note: IRRUT400 is not intended to be used for merging or rearranging databases 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.

You should run IRRUT400 when your system has little RACF activity.

How IRRUT400 Works

IRRUT400 formats and initializes each output database with an ICB, templates, segment table, BAM blocks, a complete index structure, and profiles from the input database. It also provides the following features:

- **Index compression:** IRRUT400 builds the index structure of each output database 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 databases; 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 databases:** When running with more than one input database the system prompts you for the minidisk address of the database you wish to copy.

Using IRRUT400 to Extend a Database

Use the IRRUT400 utility to copy an existing RACF database to a larger database.

By using the Split/Merge/Extend utility for an extend operation, you can:

- Compress the index to reduce the number of index blocks
- Place profile records in collating sequence near the appropriate index blocks.

Copying a RACF Database

As a general rule, running IRRUT200 is the recommended method to create a database copy.

If 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.

You may copy an active RACF database, but if you do, you must specify LOCKINPUT or guarantee that no updates will occur to the input databases from any system. Otherwise, the results of the utility and the content of the output databases will be unpredictable. If the LOCKINPUT keyword is specified, you will be unable to update the input databases after the execution of this utility.

If NOLOCKINPUT is specified and updates occur to the input databases, the results of the utility and the content of the output databases will be unpredictable.

Diagnostic Capability

IRRUT400 is not designed to provide RACF database diagnostic information. It depends on the correctness of the RACF database and may abend if it encounters corrupted data. If you suspect a RACF database error, start your problem determination by running the IRRUT200 utility described in “RACF Database-Verification Utility Program (IRRUT200)” on page 76.

See Chapter 7, “Recovery Procedures,” on page 139 and *z/VM: RACF Security Server Messages and Codes* for more information on RACF database diagnosis and correction.

Additional diagnostic information:

- IRRUT400 provides limited diagnosis when using multiple input RACF databases. In this situation, it reports inconsistencies such as duplicate profiles appearing in different data sets, or defective tape volume sets.

- If a RACF database has errors *only* in upper level blocks, you can use IRRUT400 to copy the database without copying the errors. IRRUT400 does not use the upper level blocks when copying a RACF database. Instead, it builds new upper level blocks. Therefore, if a RACF database has no errors in its profile blocks and level 1 (sequence set) blocks have no errors, the IRRUT400 utility is a good way to copy and correct the database.

Executing the Split/Merge/Extend Utility

When RACUT400 executes, you are prompted for the following:

- The option to split, merge, or copy
- The virtual address of the minidisks that contain the input and output databases
- The name of a range table (ICHRRNG) and the LOADLIB where it is located (split and merge options)
- The parameters that control IRRUT400 processing.

If a RACF database is RACF-protected, you must have at least READ authority to the database in order to issue RACUT400.

FBA Devices

For z/VM FBA devices, you *cannot* copy to a smaller database; you *can* copy to the same size database. To increase the size of a database on an FBA device, you must reallocate and rebuild the the database. If your installation is using FBA DASD, use FORMAT (for example, `FORMAT xxx z (BLK 4096)`) before executing RACINITD.

Splitting a Database

If you use the RACUT400 EXEC to split a database, RACF for z/VM allows you to have up to four databases. IBM recommends the following process:

- Split a single database into work databases.
- Redefine the work databases as the primary databases.
- Copy the primary databases to the backup databases.

Use the following procedure to split your primary and backup RACF databases.

Notes:

1. You cannot split a database that resides on an FBA device.
2. Your BACKUP databases are not actually split, but copied.
3. There must be no common disks between the input and output disk sets for SPLIT and COPY.
4. Disks created should not have any overlapping cylinder ranges to avoid conflicts.
5. The maximum number of output disks allowed for SPLIT is four.
6. Minidisks in the input and output disk sets must belong to the same group for SPLIT.

1. Log on to the RACF service machine.
2. IPL the 490 minidisk to deactivate RACF.
3. Define the work and backup databases to CP.
4. Update the database name table to reflect the number of databases.
5. Create the database range table to reflect what profiles will belong to which database.
6. Assemble and link-edit the tables.
7. Update the RACSETUP EXEC to reflect the split primary, split or copied backup, and work databases.
8. Use the RACDSF, RACALLOC, and RACINITD EXECs to prepare the work databases. Run the EXECs in that order.

9. Use the RACDSF, RACALLOC, and RACINITD EXECs to prepare the backup databases. Run the EXECs in that order.
10. Run RACUT400 to SPLIT the database into multiple databases.

    Non-interactive method for splitting databases:
    You can use the following RACUT400 syntax to accomplish this non-interactively:

    ```
    RACUT400 SPLIT input_address output_address1 output_address2 output_address3
    output_address4 (input_parameter1 input_parameter2
    ```

    The parameters in this syntax have the following functions:

    SPLIT
    Splits the input RACF database into multiple (up to four) output databases.

    input_address
    Specifies the device address of the database to be split.

    output_address1
    Specifies the device address at which the first partition of the database is to reside.

    output_address2
    Specifies the device address at which the second partition of the database is to reside.

    output_address3
    Specifies the device address at which the third partition of the database is to reside.

    output_address4
    Specifies the device address at which the fourth partition of the database is to reside.

    input_parameter
    Specifies one of the following parameters:

    LOCKinput/NOLockinput/UNLockinput
    TABLE(*table-name*)/NOTABLE
    FREESPACE(*percent*)/NOFREEspace
    ALign/NOALign
    DUPdatasets/NODUPdatasets

    Note: One of the LOCKinput/NOLockinput/UNLockinput keywords is required for SPLIT. Additional input parameters can also be specified. For descriptions of these parameters, see "Allowable Keywords" on page 94.

    The following non-interactive example splits the database at input device address 200 into four partitions at output device addresses 400, 411, 412, and 413:

    ```
    RACUT400 SPLIT 200 400 411 412 413 (NOLOCKINPUT
    ```

    Interactive method for splitting databases:
    Alternatively, you can accomplish this interactively by entering RACUT400 without parameters and then responding to system prompts.
11. Run RACUT400 to COPY the split databases to the backup databases.

    Non-interactive method for copying a database to a backup database:
    You can use the following RACUT400 syntax to accomplish this non-interactively:

    ```
    RACUT400 COPY input_address output_address (input_parameter1 input_parameter2
    ```

    The parameters in this syntax have the following functions:

    COPY
    Copies the input RACF database from the input device address to the output device address.

    input_address
    Specifies the device address of the database to be copied.

output_address
Specifies the device address to which the database is to be copied.

input_parameter
Specifies one of the following parameters:

LOCKinput/NOLockinput/UNLockinput
FREESPACE(percent)/NOFREEspace
ALign/NOALign
DUPdatasets/NODUPdatasets

Note: One of the LOCKinput/NOLockinput/UNLockinput keywords is required for COPY. Additional input parameters can also be specified. For descriptions of these parameters, see "Allowable Keywords" on page 94.

The following non-interactive example shows how to COPY the split databases to the backup databases: 400 is copied to 300, 411 is copied to 311, 412 is copied to 312, and 413 is copied to 313.

```
RACUT400 COPY 400 300 (NOLOCKINPUT
RACUT400 COPY 411 311 (NOLOCKINPUT
RACUT400 COPY 412 312 (NOLOCKINPUT
RACUT400 COPY 413 313 (NOLOCKINPUT
```

Interactive method for copying a database to a backup database:
Alternatively, you can accomplish this interactively by entering RACUT400 and responding to system prompts.

12. Start RACF and test the new databases.
13. Update the CP directory to make permanent changes.

Merging databases

RACUT400 can be used to MERGE up to four RACF databases into one RACF database.

Notes:

1. There must be no common disks between the input and output disk sets for MERGE.
2. Disks created should not have any overlapping cylinder ranges to avoid conflicts.
3. The maximum number of input disks allowed for MERGE is four.
4. Minidisks in the input and output disk sets must belong to the same group for MERGE.

Non-interactive method for merging databases:
You can use the following RACUT400 syntax to accomplish this non-interactively:

```
RACUT400 MERGE input_address1 input_address2 input_address3 input_address4output_address
(input_parameter1 input_parameter2
```

The parameters in this syntax have the following functions:

MERGE
Merges the input RACF databases from the input device addresses into a single RACF database at the output device address.

input_address1
Specifies the device address of the first database to be merged.

input_address2
Specifies the device address of the second database to be merged.

input_address3
Specifies the device address of the third database to be merged.

input_address4
Specifies the device address of the fourth database to be merged.

output_address
Specifies the device address for the merged database.

input_parameter
Specifies one of the following parameters:

LOCKinput/NOLockinput/UNLockinput
TABLE(*table-name*)/NOTABLE
FREESPACE(*percent*)/NOFREEspace
ALign/NOALign
DUPdatasets/NODUPdatasets

Note: One of the LOCKinput/NOLockinput/UNLockinput keywords is required for MERGE. Additional input parameters can also be specified. For descriptions of these parameters, see "Allowable Keywords" on page 94.

The following non-interactive example merges the databases at input device addresses 400, 411, 412, and 413 into a single database at output address 200:

```
RACUT400 MERGE 400 411 412 413 200 (NOLOCKINPUT ALIGN
```

Interactive method for merging databases:
Alternatively, you can accomplish this interactively by entering RACUT400 and responding to system prompts.

Allowable Keywords

LOCKinput/NOLockinput/UNLockinput
One of these keywords is required.

LOCKinput does not allow updates to be made to the specified input data sets, even after the utility terminates. Statistics are updated, however.

Attention:
When using LOCKinput against an active database, do not schedule maintenance spanning midnight. If the RACF database remains locked past midnight, users will not be able to submit new jobs or to 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 databases. NOLockinput is intended to be used for completely inactive RACF databases or active RACF databases in which the system and all systems sharing the databases have been completely quiesced.

If NOLockinput is specified and updates occur to the input databases, the results of the utility and the content of the output databases will be unpredictable.

UNLockinput may 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.

TABLE(*table-name*)/NOTABle
This keyword permits the specification of a user-written range table to be used to select an output database for each profile. Specifying TABLE(*table-name*) indicates that the named load module is to be used. NOTABle is the default; either specifying or defaulting to it forces the selection of OUTDD1 for all profiles.

If you are using the split or merge option, you must provide a range table (ICHRRNG) to indicate on which database to place the profiles. The information in the range table must correspond with

the information in the database name table (ICHRDSNT). For further details, see Chapter 3, “RACF Customization,” on page 19.

FREESPACE(*percent*)/NOFREEspace
This keyword allows you to control the amount of free space left in index blocks created for the output databases. 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.

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 is generally only used on z/OS. It allows you to control the processing of DATASET entries with identical names from different input databases. 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 databases, the utility copies the DATASET entry from the input database identified by the lowest-numbered ddname. When NODUPdatasets is in effect, duplicates occurring on a single input database are all accepted, assuming that they do not conflict with an entry from another database earlier in the selection sequence. NODUPdatasets is the default.

Processing of Conflicts and Inconsistencies

Do not use IRRUT400 to merge databases from different systems. If you attempt to do so, and the input databases contains duplicate user, group, or connect profiles with different contents, the output database will contain inconsistencies.

If a logical error exists in an input RACF database, 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 database.

Examples of IRRUT400 usage

These examples show how to use the split/merge/extend utility (IRRUT400) to perform database copy and split functions.

If an installation can put in place procedural controls that guarantee that the RACF database will not be moved unless an RVARY INACTIVE command is issued, the installation may choose to make the RACF data sets movable. The installation assumes the risk if the procedural controls fail.

Copying a RACF database to a larger volume without shutting down the RACFVM server

Note: This example assumes that you have a single primary database at address 200 and no work database at address 400 has been defined.

To copy the primary database (on the 200 disk) to the larger volume (on the 400 disk) using IRRUT400, logon to RACMAINT or any userid with the authority to link RACFVM's 305 disk and the database disks in write mode. Enter the following to link the disks if not already available :

1. Enter `LINK RACFVM 305 305 RR`. Enter read password if required.

2. Enter `ACCESS 305 B`.
3. Enter `LINK RACFVM 200 200 MW`. Enter multi-write password if required.
4. Define a 400 disk larger than the 200 disk) to your userid using Dirmaint or in the CP directory.
5. DSF Initialize the 400 disk using RACDSF. On the Primary disks line, enter `400` under VADDR and RACF undel LABEL.
6. Allocate a RACF dataset on the 400 disk using RACALLOC. Enter 400 to the RACALLOC prompt.
7. Execute RACUT400 to perform the copy as follows.

   Non-interactive method for copying a database:
   You can perform the copy operation non-interactively by entering the command shown in this example:

   ```
   RACUT400 COPY 200 400 (LOCKINPUT
   ```

   Multiple input parameters can be specified. For descriptions of these parameters, see “Allowable Keywords” on page 94.

   Interactive method for copying a database:
   Alternatively, you can accomplish this interactively by entering the RACUT400 command without parameters and then responding to system prompts. Like the non-interactive example, the following interactive example copies the database from input device address 200 to output device address 400:

   a. Enter `RACUT400`
   b. Reply `COPY` to the 'Enter SPLIT or MERGE or COPY or QUIT' prompt.
   c. Reply `200` to the 'Enter the single input device address' prompt.
   d. Reply `400` to the 'Enter the single output device address' prompt.
   e. Reply `YES` to the 'Do you wish to continue?' prompt.
   f. Reply `CONT` to enter parameters.
   g. Reply `LOCKINPUT` to the first 'Enter Next Parameter' prompt.
   h. Reply `END` to the second 'Enter Next Parameter' prompt.

   Return code from 'IRRUT400' = 0 should be issued if successful.

You will now have a larger copy of the RACF primary database on the 400 disk. The 200 disk (RACF primary) will be locked from updates as the LOCKINPUT parm was used during the copy. For example, if an ALTUSER password update is attemped you will get the following message:

```
ICH51 7I RACF DATA SET CANNOT BE ALTERED. IT HAS BEEN EXTENDED BY RACF
```

To unlock the RACF primary, run IRRUT400 again specifying UNLOCKINPUT, as follows:

1. Enter `RACUT400`
2. Reply `COPY` to the 'Enter SPLIT or MERGE or COPY or QUIT' prompt.
3. Reply `200` to the 'Enter the single input device address' prompt.
4. Reply `200` to the 'Enter the single output device address' prompt.
5. Reply `YES` to the 'Do you wish to continue?' prompt.
6. Reply `CONT` to enter parameters.
7. Reply `UNLOCKINPUT` to the first 'Enter Next Parameter' prompt.
8. Reply `END` to the second 'Enter Next Parameter' prompt.
9. Return code from 'IRRUT400' = 0 should be issued if successful.

The following procedure may be used to get the RACFVM server to use the larger database on the 400 disk as its primary database. This example assumes you are using RACMAINT as your authorized user and RACFVM is your RACF server.

1. From RACMAINT, switch from primary to backup database by entering command `RAC RVARY SWITCH`. You may need to enter a password from the VM Operator console to confirm RVARY actions.
2. Link or attach the 400 disk to RACFVM.
3. From RACFVM, detach the current 200 disk. Enter `#CP DET 200`.
4. From RACFVM, define the 400 disk as 200. Enter `#CP DEF 400 200`.
5. From RACMAINT, activate the primary dataset by entering command `RAC RVARY ACTIVE DA(RACF.DATASET)`.
6. From RACMAIN, switch from backup to primary database by entering command `RAC RVARY SWITCH`.

Note: There is a window in between the first RVARY SWITCH (to activate the backup database) and the second (to activate the new primary) during which updates to the primary database are not possible. Any updates made at this time (which will be to the backup database) will be lost when the new primary database is activated. It is strongly recommended that this process be undertaken at a time of low RACF activity.

You will need to make CP directory changes if the 200/400 minidisk swap is to be permanent.

Splitting One Database into Four Databases

Step One

Note: RACF for z/VM allows you to have up to four databases, which should belong to the following:

- 200, 211, 212, and 213 as the primary databases
- 300, 311, 312, and 313 as backup databases
- 400, 411, 412, and 413 as the work databases

First define the work databases in the CP directory, using virtual addresses 400, 411, 412, and 413:

1. Specify the temporary disk and number of cylinders, using the **`def t3390`** *`disk cylinders`* command, where *`disk`* is replaced by the work disks used by RACF (400, 411, 412, and 413). For example:

   ```
   def t3390 400 200
   ```

2. Format the disk file mode, as shown in this example:

   ```
   format 400 E
   ```

   Enter "1" to format the disk, then enter the name of the temp disk (preferably TMP400).

Next define the backup databases in the CP directory, using virtual addresses 300, 311, 312, and 313.

Note: If you already have a 300, simply change the CP directory entry to another address (for example, change 300 to 333).

CAUTION: Do not place any of these minidisks on cylinder 0 in the CP directory, because when the RACDSF EXEC executes, it could destroy the volume identifier on the pack.

Step Two

1. Log on to the appropriate **`MAINT`***`vrm`* user ID (for example, MAINT730 or MAINT740).
2. Access the service disks:

   ```
   VMFSETUP SERVP2P compname
   ```

   For *`compname`*, use:

   RACF
   For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

Example of ICHRDSNT ASSEMBLE—One Database

```
ICHRDSNT CSECT
*********************************************************************
*        ICHRDSNT - RACF DATA SET NAME TABLE
*********************************************************************
         SPACE 2
*********************************************************************
*      NUMBER OF NAME PAIR ENTRIES
*********************************************************************
         SPACE 2
         DC    X'01'
         SPACE 2
*********************************************************************
*        DATA SET NAMES
*        FORMAT:
*              CL44'PRIMARY DSNAME',CL44'SECONDARY DSNAME',X'N1,X'N2'
*
*                    N1=NUMBER OF RESIDENT BLOCKS
*                    N2=FLAG FIELD
*                         BIT 0 = UPDATES ARE TO BE DUPLICATED
*                         BIT 1 = STATISTICS ARE TO BE DUPLICATED
*                         BIT 7 = USE RESIDENT DATA BLOCK OPTION
*
*
*********************************************************************
         SPACE   3
*********************************************************************
NAME1    DC    CL44'RACF.DATASET',CL44'RACF.BACKUP',X'64',X'81'
*********************************************************************
         END   ICHRDSNT
```

Example of ICHRDSNT ASSEMBLE—Four Databases

Note the changes marked with a “>”.

```
ICHRDSNT CSECT
*********************************************************************
*        ICHRDSNT - RACF DATA SET NAME TABLE
*********************************************************************
         SPACE 2
*********************************************************************
*      NUMBER OF NAME PAIR ENTRIES
*********************************************************************
         SPACE 2
>        DC    X'04'
         SPACE 2
*********************************************************************
*        DATA SET NAMES
*        FORMAT:
*              CL44'PRIMARY DSNAME',CL44'SECONDARY DSNAME',X'N1,X'N2'
*
*                    N1=NUMBER OF RESIDENT BLOCKS
*                    N2=FLAG FIELD
*                         BIT 0 = UPDATES ARE TO BE DUPLICATED
*                         BIT 1 = STATISTICS ARE TO BE DUPLICATED
*                         BIT 7 = USE RESIDENT DATA BLOCK OPTION
*
*
*********************************************************************
         SPACE   3
*********************************************************************
NAME1    DC    CL44'RACF.DATASET',CL44'RACF.BACKUP',X'64',X'81'
> NAME2    DC    CL44'RACF.DATASET1',CL44'RACF.BACKUP1',X'64',X'81'
> NAME3    DC    CL44'RACF.DATASET2',CL44'RACF.BACKUP2',X'64',X'81'
> NAME4    DC    CL44'RACF.DATASET3',CL44'RACF.BACKUP3',X'64',X'81'
```

```
*********************************************************************
         END   ICHRDSNT
```

You need to:

- Modify the ICHRDSNT ASSEMBLE file to change the database count from 1 to 4.
- Assemble ICHRDSNT ASSEMBLE after making the changes.
- Link-edit the new ICHRDSNT TEXT into RACFLINK LOADLIB.

Follow the instructions in "Modify Full-Part ASSEMBLE and TEXT Files" on page 160, using the following substitution values. Note that you are building the RACFOBJ TXTLIB and RACFLINK LOADLIB libraries.

- For *fn* use **ICHRDSNT**
- For *nnnn* use **0002**

Step Three

Create a new file called ICHRRNG ASSEMBLE to reflect what RACF profiles belong to which of the four databases. Take into account your installation's profile-naming conventions so that you optimize your database organization. For example, you can choose to put all profiles that range from A-D into the first database, E-H into the second database, I-P into the third database, and Q-Z into the fourth database.

Remember, the class name is prefixed to the profile name. For instance, if you have a minidisk profile called USER1.191, the actual profile name is VMMDISK.USER1.191.

Example of ICHRRNG ASSEMBLE File—One Database

```
ICHRRNG   CSECT
          DC      F'1'
PART1     DC      XL44'00'
          DC      AL1(1)
          END     ICHRRNG
```

Example of ICHRRNG ASSEMBLE File—Four Databases

```
ICHRRNG   CSECT
          DC      F'4'
PART1     DC      XL44'00'
          DC      AL1(1)
PART2     DC      XL1'C5',XL43'00'
          DC      AL1(2)
PART3     DC      XL1'C9',XL43'00'
          DC      AL1(3)
PART4     DC      XL1'D8',XL43'00'
          DC      AL1(4)
          END     ICHRRNG
```

You need to:

- Create or modify the ICHRRNG ASSEMBLE file.
- Assemble ICHRRNG ASSEMBLE.
- Link-edit the new ICHRRNG TEXT into RACFLPA LOADLIB.

Follow the instructions in "Modify Full-Part Replacement TEXT Files" on page 163, using the following substitution values. Note that you are building the RACFOBJ TXTLIB and RACFLPA LOADLIB libraries.

- For *fn* use **ICHRRNG**
- For *nnnn* use **0002**

Step Four

Edit the RACSETUP EXEC to uncomment the multiple extents for split and work databases. To define multiple extents for split or work databases, remove the comment indicators from the following lines in the ACCESS section and the FILEDEF section:

```
/*   "AC" racfp1 primode1                        */
/*   ds.2 = RC                                   */
/*   "AC" racfp2 primode2                        */
/*   ds.3 = RC                                   */
/*   "AC" racfp3 primode3                        */
/*   ds.4 = RC                                   */
/*   "AC" racfbk1 bkmode1                        */
/*   dsb.2 = RC                                  */
/*   "AC" racfbk2 bkmode2                        */
/*   dsb.3 = RC                                  */
/*   "AC" racfbk3 bkmode3                        */
/*   dsb.4 = RC                                  */
/*   "AC" work wkmode                            */
/*   "AC" work1 wkmode 1                         */
/*   "AC" work2 wkmode 2                         */
/*   "AC" work3 wkmode 3                         */
/*   "FI" ddf bkmode1 "DSN" racf_backup1         */
/*   retc = MAX(RC,dsb.2,retc)                   */
/*   "FI" ddg bkmode2 "DSN" racf_backup2         */
/*   retc = MAX(RC,dsb.3,retc)                   */
/*   "FI" ddh bkmode3 "DSN" racf_backup3         */
/*   retc = MAX(RC,dsb.4,retc)                   */
/*   "FI" ddb primode1 "DSN" racf_dataset1       */
/*   retc = MAX(RC,dsb.2,retc)                   */
/*   "FI" ddc primode2 "DSN" racf_dataset2       */
/*   retc = MAX(RC,dsb.3,retc)                   */
/*   "FI" ddd primode3 "DSN" racf_dataset3       */
/*   retc = MAX(RC,dsb.4,retc)                   */
/*   "FI" ddb bkmode1 "DSN" racf_backup1         */
/*   retc = MAX(RC,dsb.2,retc)                   */
/*   "FI" ddc bkmode2 "DSN" racf_backup2         */
/*   retc = MAX(RC,dsb.3,retc)                   */
/*   "FI" ddd bkmode3 "DSN" racf_backup3         */
/*   retc = MAX(RC,dsb.4,retc)                   */
```

Follow the instructions in “Modify Full-Part Replacement Parts” on page 165, using the following substitution values:

- For *fn* use **RACSETUP**
- For *ft* use **EXEC**
- For *nnnn* use **0002**

Step Five

1. Logon to the RACF service machine.
2. IPL the 490 minidisk.
3. Run the RACDSF, RACALLOC, and RACINITD EXECs to OS-format, allocate, and initialize the work databases.

Note: RACDSF, RACALLOC, and RACINITD execs can be used with both IPL CMS and IPL 490.

The RACDSF EXEC

The RACDSF EXEC formats each work database (400, 411, 412, and 413). The EXEC is interactive. Follow the prompts during execution.

Enter: RACDSF

The following screen appears:

```
                    RACF/VM - OS Formatting Utility        Screen 1 of 1

 To OS format minidisks, type the virtual addresses and labels and press PF2.
 Default labels will be used for labels not specified.

                   VADDR  LABEL
   Primary disks   ____ | ______
                   ____ | ______
                   ____ | ______
                   ____ | ______
   Backup disks    ____ | ______
                   ____ | ______
                   ____ | ______
                   ____ | ______
   Work disks      ____ | ______
                   ____ | ______
                   ____ | ______
                   ____ | ______


  PF1=Help     PF2=OS format  PF3=Exit
Enter CP/CMS Commands below
====>
```

Fill in the device addresses and optional label fields (the label should conform to your installation's naming conventions). When you complete this screen, press PF2 to OS format the minidisks.

> **Split Databases:** If you have split databases, you can enter information for all output databases consecutively, complete the screen, and press PF2.
>
> An example for four databases:
>
> - Type a device address (400) and a new-label for it (temp00).
> - Type a device address (411) and a new-label for it (temp11).
> - Type a device address (412) and a new-label for it (temp12).
> - Type a device address (413) and a new-label for it (temp13).
> - Press PF2.

When the formatting completes, you will receive a message.

The RACALLOC EXEC allocates a database. When you are prompted for the database, enter the virtual address for the CUU (example, 400). Invoke the RACALLOC EXEC for each database.

Example of the RACALLOC EXEC:

Enter:	RACALLOC	
Enter:	400	for CUU.
Enter:	YES	for a database.
or		
Enter:	NO	for a non database.

Reinvoke RACALLOC for the 411, 412, and 413 databases.

Note: You cannot split a non database into a database. You must use what your current database is.

Determine the type of database by checking the block size:

```
Enter: ACC 200 E
   LISTDS E (FO
```

If the BLKSI is 4096, it is . If the BLKSI is 1024, it is not .

Run RACALLOC for each additional database — 411, 412, and 413.

The RACINITD EXEC

The RACINITD EXEC initializes a database. When you are prompted for the database, enter the virtual address for the CUU (for example, 400). You must invoke the RACINITD EXEC for each database.

An example of the RACINITD EXEC:

```
Enter: RACINITD
Enter: 400      for CUU
```

At this point:

400 is set up as RACF.DATASET
411 is set up as RACF.DATASET1
412 is set up as RACF.DATASET2
413 is set up as RACF.DATASET3.

You can verify this by entering: `ACC xxx E`, where *xxx* is the virtual address of the database and *E* is any free access mode. Then enter `LISTDS E (FO`.

Step Six

Run the RACDSF, RACALLOC, and RACINITD EXECs to OS-format, allocate, and initialize the backup databases (300, 311, 312, and 313).

Follow the detailed instructions you followed in Step Five.

When you run the RACDSF EXEC, use your own naming conventions for the new-label field. After RACINITD executes:

300 is set up as RACF.BACKUP
311 is set up as RACF.BACKUP1
312 is set up as RACF.BACKUP2
313 is set up as RACF.BACKUP3.

Step Seven

Run the RACUT400 EXEC to split the database into multiple databases. The EXEC is interactive; simply follow the prompts.

Example of the RACUT400 EXEC (Splitting Four Databases)

Note: Installations can enter any valid keyword—NOLOCKINPUT was chosen for the example.

```
Enter:   IPL 490       to stop RACF.
Enter:   RACUT400
Press ENTER to continue.
Enter:   SPLIT         to split the database into multiple extents.
Enter:   200           for the input database.
Enter:   400           for one of the output databases.
Enter:   411           for one of the output databases.
Enter:   412           for one of the output databases.
Enter:   413           for one of the output databases.
Press ENTER to end the output databases.
Enter:   ICHRNNG       for the range table name.
Enter:   RACFLPA       for the load library name.
Enter:   YES           to continue.
Enter:   CONT          for input keywords.
(See "Allowable Keywords" on page 94 or enter HELP.)
Enter:   NOLOCKINPUT
Enter:   END           to end the keywords.
```

A return code of 0 means successful execution. All output from the RACUT400 EXEC will be placed in UT400 OUTPUT on the A-disk. If you get a nonzero return code, look at UT400 OUTPUT for the error indication.

Step Eight

Run the RACUT400 EXEC to COPY the work databases to the backup databases.

Example of the RACUT400 EXEC (Copying the Backup Databases)

Note: Installations can enter any valid keyword—NOLOCKINPUT was chosen for the example.

```
Enter:   RACUT400
Press ENTER to continue.
Enter:   COPY          to copy the 200 database to the 300 database.
Enter:   400           for the input database.
Enter:   300           for the output database.
Enter:   YES           to continue.
Enter:   CONT          for input keywords.
(See "Allowable Keywords" on page 94 or enter HELP.)
Enter:   NOLOCKINPUT
Enter:   END           to end the keywords.
```

A return code of 0 means successful execution. All output from the RACUT400 EXEC will be placed in UT400 OUTPUT on the A-disk. If you get a nonzero return code, look at UT400 OUTPUT for the error indication.

```
Enter:   RACUT400
Press ENTER to continue.
Enter:   COPY          to copy the 211 database to the 311 database.
Enter:   411           for the input database.
Enter:   311           for the output database.
Enter:   YES           to continue.
Enter:   CONT          for input keywords.
(See "Allowable Keywords" on page 94 or enter HELP.)
Enter:   NOLOCKINPUT
Enter:   END           to end the keywords.
```

A return code of 0 means successful execution. All output from the RACUT400 EXEC will be placed in UT400 OUTPUT on the A-disk. If you get a nonzero return code, look at UT400 OUTPUT for the error indication.

```
Enter:   RACUT400
Press ENTER to continue.
Enter:   COPY          to copy the 212 database to the 312 database.
Enter:   412           for the input database.
Enter:   312           for the output database.
Enter:   YES           to continue.
Enter:   CONT          for input keywords.
(See "Allowable Keywords" on page 94 or enter HELP.)
Enter:   NOLOCKINPUT
Enter:   END           to end the keywords.
```

A return code of 0 means successful execution. All output from the RACUT400 EXEC will be placed in UT400 OUTPUT on the A-disk. If you get a nonzero return code, look at UT400 OUTPUT for the error indication.

```
Enter:  RACUT400
Press ENTER to continue.
Enter:  COPY          to copy the 213 database to the 313 database.
Enter:  413           for the input database.
Enter:  313           for the output database.
Enter:  YES           to continue.
Enter:  CONT          for input keywords.
(See "Allowable Keywords" on page 94 or enter HELP.)
Enter:  NOLOCKINPUT
Enter:  END           to end the keywords.
```

A return code of 0 means successful execution. All output from the RACUT400 EXEC will be placed in UT400 OUTPUT on the A-disk. If you get a nonzero return code, look at UT400 OUTPUT for the error indication.

Step Nine

Test the new databases by temporarily redefining the single database to a different virtual address and redefining the work databases to 200, 211, 212, and 213.

Note: The production virtual addresses 200, 211, 212, and 213 are those shipped in the RACSETUP and RACONFIG EXECs. They are the addresses that the RACF service machine is expecting at initialization. The work virtual addresses 400, 411, 412, and 413 are shipped in the RACSETUP and RACONFIG EXECs.

Example of Temporarily Redefining the Databases for Testing

```
Enter: define 200 222 (where 222 is not known to RACF for z/VM).
Enter: define 400 200
Enter: define 411 211
Enter: define 412 212
Enter: define 413 213
```

Note: At this point, there are four production databases at addresses 200, 211, 212, and 213 and four backup databases at addresses 300, 311, 312, and 313.

Bring up RACF to test the new databases:

```
Enter: IPL 490
Enter: RACSTART   to start RACF.
```

Step Ten

Splitting your database is now complete. However, you should make these databases permanent. In order for RACF to use the databases, they must reside on the 200, 211, 212 and 213 minidisks. You may achieve this by using either of the following options:

- Define the virtual addresses your database resides on (in this example, 400, 411, 412, and 413) to virtual addresses 200, 211, 212 and 213 in the CP directory.
- Copy the database. Using the DASD DUMP RESTORE (DDR) command, copy the database residing at virtual addresses 400, 411, 412, and 413 to virtual addresses 200, 211, 212, and 213.

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)	Duplicate IBM-defined names caused one or more warning conditions.

Hex	(Decimal)	Meaning
8	(8)	One or more error conditions occurred because of one of the following conditions: • Duplicate non-IBM-defined names • A defective tape-volume set.
C	(12)	One or more severe error conditions resulted from an error on an output database.
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 database.

Chapter 6. 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 offered by RACF, as well as to optimize its usability. For RACROUTE requests, the exits allow an installation to tailor the parameters passed on the macro instruction and to perform any additional security checks or processing that the installation requires.

The exit routines must be reenterable and refreshable and must be located in the link-pack area: RACFLPA or RACFLINK LOADLIB. 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

When the preprocessing exit routines receive control, RACF has already validity-checked the macro instruction parameters, but has not yet performed any other processing.

Make changes or additions to the parameter information only in the designated areas. 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.

See *z/VM: Security Server RACROUTE Macro Reference* 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, 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.

Note: You must have the AUDITOR or ROAUDIT attribute to run the report. See *z/VM: RACF Security Server 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.

© Copyright IBM Corp. 1993, 2025

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.

Possible Uses of RACF Exits

Some possible uses of the RACF exits are described with the individual exit. They are:

- "ICHPWX01 Use Case: Password Quality Control" on page 115
- Password phrase quality control. See "New-Password-Phrase Exit" on page 115.
- Allowing resource access for alternate user IDs under limited circumstances. See "RACROUTE REQUEST=AUTH Exits" on page 118.

Summary of Installation-Exit Callers

Table 4 on page 108 and Table 5 on page 109 list the macros, commands, and utilities that give control to each exit routine.

Table 4. RACF Installation-Exits Cross-Reference Table—Part 1 of 2

Functions	Data Set Naming Conventions Table ICHNCV00	RACDEF Pre- and Post- ICHRDX01, 02	RACHECK Pre- and Post- ICHRCX01, 02	RACINIT Pre- and Post- ICHRIX01, 02
ADDSD	X (Note 1, Note 2)	Note 1	Note 2	
ALTDSD			Note 2	
ALTUSER				
DELDSD	X (Note 1, Note 2)	Note 1	Note 2	
DELGROUP				
DELUSER				
LISTDSD	X (Note 2)		Note 2	
PASSWORD				
PERMIT	X (Note 2)		Note 2	
RALTER			Note 2	
RDEFINE		Note 1	Note 2	
RDEFINE FROM (on data sets)	X (Note 1, Note 2)	Note 1	Note 2	
RDELETE		Note 1	Note 2	
REMOVE				
RLIST			X	

Table 4. RACF Installation-Exits Cross-Reference Table—Part 1 of 2 (continued)				
Functions	**Data Set Naming Conventions Table ICHNCV00**	**RACDEF Pre- and Post- ICHRDX01, 02**	**RACHECK Pre- and Post- ICHRCX01, 02**	**RACINIT Pre- and Post- ICHRIX01, 02**
SEARCH	X		X	Note 3
SETROPTS RACLIST				Note 3
RACDEF	X	X	Note 2	Note 3
RACHECK	X		X	Note 3
RACINIT				X
RACLIST				
FRACHECK				
IRRUT100	X			
RACXTRT			Note 2	
SIGNON				Note 3

Notes

1.
The function invokes RACDEF processing and may be affected by the RACDEF exits.

2.
The function may invoke RACHECK processing and may be affected by the RACHECK exits.

3.
The function may invoke RACINIT processing and may be affected by the RACINIT exits.

Table 5. RACF Installation-Exits Cross-Reference Table—Part 2 of 2						
Function	**RACLIST Pre-, Post-, and Selection ICHRLX01, 02**	**FRACHECK Pre- and Post- ICHRFX01, 02**	**New Password ICHPWX01**	**New Password Phrase ICHPWX11**	**Command ICHCNX00**	**Command ICHCCX00**
ADDSD					X	
ADDUSER				X		
ALTDSD					X	
ALTUSER			X	X		
DELDSD					X	
DELGROUP						X
DELUSER						X
LISTDSD					X	
PASSWORD			X			
PERMIT					X	
RALTER	Note 4	Note 5				
RDEFINE	Note 4	Note 5			X	

Table 5. RACF Installation-Exits Cross-Reference Table—Part 2 of 2 (continued)						
Function	**RACLIST Pre-, Post-, and Selection ICHRLX01, 02**	**FRACHECK Pre- and Post- ICHRFX01, 02**	**New Password ICHPWX01**	**New Password Phrase ICHPWX11**	**Command ICHCNX00**	**Command ICHCCX00**
RDEFINE FROM (on data sets)						
RDELETE						
REMOVE						X
RLIST						
SEARCH					X	
SETROPTS RACLIST	Note 6					
RACDEF						
RACHECK						
RACINIT			X	X		
RACLIST	X					
FRACHECK		X				
IRRUT100					X	
RACXTRT					X	
SIGNON						

Notes

4. When the user is not SPECIAL and has specified ADDMEM and DELMEM, this function invokes RACLIST processing and may be affected by the RACLIST exits.

5. When the user is not SPECIAL and has specified ADDMEM and DELMEM, the function invokes FRACHECK and may be affected by the FRACHECK exits.

6. The function invokes RACLIST processing and may be affected by the RACLIST exits.

RACROUTE REQUEST=VERIFY(X) Exits

The term RACINIT originally referred to the independent system macro. The RACINIT macro was replaced by RACROUTE REQUEST=VERIFY and RACROUTE REQUEST=VERIFYX. We recommend that customers use RACROUTE requests in their applications. Many customers are familiar with the term RACINIT, and we continue its use in the RACF library.

A RACINIT request is used to determine whether a user ID is defined to RACF and whether the user has supplied a valid password and group name. During z/VM logon processing, RACINIT also determines whether a user is authorized to access the terminal.

If the user ID, password or password phrase, group name, and terminal are accepted, RACF builds an access-environment element (ACEE) for the user.

Note: When no user ID, group, and password are passed to RACINIT, RACINIT builds a default ACEE containing an asterisk (*) (X'5C') for the user ID and group name and returns to the issuer of RACINIT 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. You can use the RACINIT 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 RACINIT preprocessing and

postprocessing exits are derived from the parameters specified on the RACROUTE macro instruction. For more details, see *z/VM: Security Server RACROUTE Macro Reference*.

Preprocessing Exit (ICHRIX01)

The RACINIT preprocessing exit routine must be named ICHRIX01. It is entered before:

- User identification
- User verification
- Terminal authorization checking

This exit must be reentrant and is invoked in supervisor state, under protection key 0, with no locks held.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHRIX01**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRIX01**

The ICHRIXP macro maps the RACINIT exit parameter list. To find a mapping of RIXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

When you delete an ACEE that has a third-party ACEE attached, the RACINIT preprocessing and postprocessing exits get control again, both for the third-party ACEE and for the original ACEE being deleted. This allows explicit access to the installation work area's ACEEIEP field off any third-party ACEEs.

RACINIT 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 will be nested.

For example, the preprocessing exit will be called for the main ACEE. Then another RACROUTE REQUEST=VERIFY 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 RACINIT Preprocessing Exit

When the RACINIT 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.

8
The request is accepted. Processing stops, but the postprocessing exit is still invoked.

Note: If register 15 contains any other value, RACINIT issues an abend code (383) that indicates a nonvalid exit return code.

Do not confuse codes from the RACINIT preprocessing exit routine with the return codes from the RACROUTE REQUEST=VERIFY(X) macro, which are documented in *z/VM: Security Server RACROUTE Macro Reference*.

When the RACINIT preprocessing exit routine sets a return code of 8 and then issues a RACINIT macro-instruction request with ENVIR=CREATE, the exit routine creates and initializes the access-control environment element (ACEE). If you are using an exit routine:

- Your exit routine may use the ACEE passed as input, or it may obtain its own storage for the ACEE. If the exit routine obtains its own storage, the exit must free the ACEE and the tables chained to it.
- If the exit routine builds its own ACEE, the ACEE must conform to the standard mapping of the control block, because all of the RACF service routines and commands depend on that format.

Postprocessing Exit (ICHRIX02)

The RACINIT postprocessing exit routine must be named ICHRIX02. It is entered after user identification and verification and terminal authorization checking.

This exit must be reentrant and invoked in supervisor state, with protection key 0, with no locks held.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHRIX02**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRIX02**

When the RACINIT postprocessing exit routines receive control, RACF has already performed the main function (for example, ACEE creation), but has not performed any logging or statistics recording.

The ICHRIXP macro maps the RACINIT exit parameter list. To find a mapping of RIXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the RACINIT Postprocessing Exit

When the RACINIT postprocessing exit routine returns control, register 15 should contain one of the following return codes:

Code
Meaning

0
Continue with RACINIT processing. If the exit routine changes the values of the return code or the abend code (from zero to a nonzero value), RACINIT will use the changed values.

4
Try the RACROUTE request again; invoke the RACINIT 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, RACINIT issues an abend code (383) that indicates a nonvalid exit return code.

The RACINIT macro instruction may have updated the user's entry with new password information. In this case, attempts to retry the RACROUTE request without adjusting the input parameters accordingly may cause RACINIT failure.

Do not confuse return codes from the RACINIT postprocessing exit routine with the return codes from the RACROUTE REQUEST=VERIFY(X) macro, which are documented in *z/VM: Security Server RACROUTE Macro Reference*.

New-Password Exit

RACINIT 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.

ICHPWX01 Processing

The new-password exit must be named ICHPWX01.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHPWX01**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHPWX01**

The ICHPWXP macro maps the new password exit parameter list. To find a mapping of PWXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

The ICHPWX01 routine is invoked in the following ways:

- **Through RACINIT processing.** If you specify a new password, RACINIT processing first invokes ICHRIX01 (if ICHRIX01 is present in the system), and then immediately invokes ICHPWX01, before checking the current password.
- **Through the ALTUSER command.** If you specify the PASSWORD keyword with a password value, ALTUSER invokes ICHPWX01 after parsing and checking the user's authorization.
- **Through the PASSWORD command.** If you specify the PASSWORD or INTERVAL keywords and the conditions listed below are met, PASSWORD invokes ICHPWX01 after parsing and checking the user's authorization:
  - The new password differs from the current password.
  - The new password differs from the previous passwords, if the password-history option is active.
  - The new password obeys all of the installation's syntax rules.

Table 6 on page 114 shows which fields are available to the exit when the exit is called from the different RACF components.

Table 6. Fields Available during ICHPWX01 Processing				
OFFSET (Decimal	**PARAMETER (Address)**	**RACINIT**	**ALTUSER**	**PASSWORD**
0	Length	X	X	X
4	Caller	X	X	X
8	Command-processor parameter list	-	X	X
12	NEWPASS	X	X	O
16	INTERVAL	-	-	O
20	User ID	X	X	X
24	Work area	X	X	X
28	Current password	X[1]	-	O[2]
32	Password is last change date	X	-	O[2]
36	ACEE	X[3]	X	X
40	Group name	O	-	-
44	Installation data	O	-	-
48	Password history	X	-	O[2]
52	Flag byte	X	-	-

Note:

1. Not available if PASSCHK=NO was specified.
2. Available only if NEWPASS is available and KDFAES is not enabled.
3. Although available, the ACEE may not be fully initialized.

Return Codes from the New-Password Exit

When the password exit routine returns control, register 15 should contain one of the following return codes:

Hex	**(Decimal)**	**Meaning**
0	(0)	The new-password field and the interval value will be copied back into the calling function. Continue with processing.
4	(4)	The new-password request is not accepted and is to be failed. RACINIT processing will terminate with a return code indicating an invalid new password. The ALTUSER command will ignore the request and continue processing. The PASSWORD command will terminate 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 RACINIT; return codes 0, 4, 12, and 16 are valid for ALTUSER, and 0, 4, 8, 12, and 16 are valid for PASSWORD. If register 15 contains any other values, processing ends with an abend.

ICHPWX01 Use Case: 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 RACINIT function. At this time the user is able (or may 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 RACINIT 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 may disallow names, months, user IDs, and group names, or detect trivial usage of alphanumeric mixes such as JAN07 and FEB07.

Using the ICHPWX01 sample exit

IBM provides an ICHPWX01 exit in source form, as file ICHPWX01 ASSEMBLE on the RACF 305 disk. This sample invokes a sample REXX exit, shipped in source form as IRRPWREX SAMPLE on the 305 disk. As shipped, the REXX exec can enforce the following checks:

- minimum length
- maximum length
- list of allowable characters
- password contains user name

These checks can be enabled or disabled by modifying configuration variables in the exec. By default, none of the checks are enabled. Running the exec is functionally equivalent to having no exit.

For more information on installing the sample ICHPWX01 exit, see “ICHPWX01 Processing” on page 113.

Copy IRRPWREX SAMPLE from the 305 to the 191, rename it to IRRPWREX EXEC, and modify it for your purpose. The RACF service machine must be restarted to activate the exit.

New-Password-Phrase Exit

RACROUTE REQUEST=VERIFY processing and the ADDUSER, ALTUSER, and PASSWORD commands invoke the installation-supplied new-password-phrase processing exit.

The installation has the option of using the new-password-phrase exit to augment RACF function when validating a new password phrase.

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 user 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 the pass 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 continues to be processed by the new password exit, ICHPWX01 and will not be passed to this exit.

ICHPWX11 processing

The new-password-phrase exit must be named ICHPWX11.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Modify Full-Part ASSEMBLE and TEXT Files" on page 160. Use the following substitution values:

- for *fn*, use ICHPWX11
- for *nnnn*, use 0002

The ICHPWX2 macro maps the new password phrase exit parameter list. To find a mapping of the exit parameter list, PWX2, look in the RACF MACLIB file on the RACF service machine's 305 disk.

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 pass 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 function:

  – Validates the new password phrase for compliance with RACF's password phrase rules
  – Verifies that the new password phrase differs from the current pass phrase
  – If the SETROPTS PASSWORD(HISTORY) option is acrive, verifies that the new password phrase differs from the previous password phrases

  If the password phrase passes these checks, the command invokes ICHPWX11.

Table 7 on page 117 shows which fields are available to the exit when the exit is called from the different RACF components.

Table 7. Fields Available during ICHPWX11 Processing				
OFFSET (Decimal	**PARAMETER (Address)**	**REQUEST= VERIFY**	**ADDUSER/ ALTUSER**	**PASSWORD**
0	Length	X	X	X
4	Caller	X	X	X
8	Command-processor parameter list	-	X	X
12	New password phrase	X	X	X
16	User ID	X	X	X
20	Work area	X	-	-
24	Current password phrase	X[1]	-	O[2]
28	password phrase last change date	X	-	O[2]
32	ACEE	X[3]	X	X
36	Group name	O	-	-
40	Installation data	O	-	-

X
means “always available.”

O
means “may be available.”

—
means “never available.”

Note:

1. Not available if PASSCHK=NO was specified.
2. Available only if new password phrase is available.
3. Although available, the ACEE may not be fully initialized.

Return Codes from the New-Password-Phrase Exit

When the password phrase exit routine returns control, register 15 should contain one of the following return codes:

Hex	**(Decimal)**	**Meaning**
0	(0)	The new password phrase field is copied back into the calling function. Continue with processing.
4	(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	(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: Decimal return codes 0 and 4 are valid for RACROUTE REQUEST=VERIFY; if register 15 contains any other values, processing ends with an abend.

Return codes 0, 4, and 8 are valid for ADDUSER, ALTUSER, and PASSWORD. If register 15 contains any other values, it is treated like return code 4.

Using the ICHPWX11 sample exit

IBM provides an ICHPWX11 exit in source form, as file ICHPWX11 ASSEMBLE on the RACF 305 disk. This sample will invoke a sample REXX exit, shipped in source form as IRRPHREX SAMPLE on the 305 disk. As shipped, the REXX exec can enforce the following checks:

- minimum length
- maximum length
- list of allowable characters
- leading blanks allowed or not
- trailing blanks allowed or not
- words in user name allowed or not
- triviality checks (new differs from old by more than just case or spaces, and one is not a substring of the other)
- minimum unique characters by position from old password phrase
- minimum unique words from old password phrase
- dictionary check (hard coded list of words)

These checks can be enabled or disabled by modifying configuration variables in the exec. By default, none of the checks will be enabled. Running the exec will be functionally equivalent to having no exit.

See "ICHPWX11 processing" on page 116 for more information on installing the sample ICHPWX11 exit.

Copy IRRPHREX SAMPLE from the 305 to the 191, rename it to IRRPHREX EXEC, and modify it as desired. The RACF service machine must be restarted in order to activate the exit.

RACROUTE REQUEST=AUTH Exits

The term RACHECK originally referred to the independent system macro. The RACHECK macro was replaced by RACROUTE REQUEST=AUTH. We recommend that customers use RACROUTE requests in their applications. Many customers are familiar with the term RACHECK, and we continue its use in the RACF library.

A RACHECK request 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, RACHECK 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 RACHECK 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.)

The ICHRCX02 exit (as shipped) allows an alternate user to access resources the service machine can access, but that the alternate user normally cannot. This sample exit was originally written to allow users to access a restricted compiler but only when they are submitting batch jobs to a specified batch machine. The requests are re-driven using the ACEE of the service machine when the request is for a resource in the VMNODE, VMRDR, or VMMDISK class. This exit is shipped but not enabled, see "Adding the ICHRCX02 Exit" on page 121 if you wish to enable it.

Many of the values passed to the exits are derived from the parameters specified on the macro instruction. For details of the RACROUTE REQUEST=AUTH macro instruction, see *z/VM: Security Server RACROUTE Macro Reference*.

Preprocessing Exit (ICHRCX01)

The 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:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHRCX01**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRCX01**

When the RACHECK preprocessing exit receives control for the DATASET class, RACF has already processed the naming conventions table, if there is one.

The ICHRCXP macro maps the RACHECK exit parameter list. To find a mapping of RCXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the RACHECK Preprocessing Exit

When the RACHECK 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, the meanings of which are documented in *z/VM: Security Server RACROUTE Macro Reference*.

When the RACHECK preprocessing exit returns a return code of 4 or 8 and the RACHECK macro-instruction request 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. RACHECK is not to perform any authorization checking on the access list, but other normal RACHECK processing (such as default return code processing, PROTECTALL processing, and logging) is to continue.

Note:

1. If register 15 contains any other value, RACHECK issues an abend code (382) that indicates a nonvalid exit return code.
2. The RACHECK 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 RACHECK preprocessing exit, see the description of the naming convention exit, ICHCNX00. The ICHCNXP macro

maps the naming convention parameter list. To find a mapping of CNXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

RACF uses resident profiles in two ways:

- As installation-supplied profiles
- As specified by an exit routine.

The ICHRRPF macro maps the resident profile.

To find a mapping of RRPF, look in the RACF MACLIB file on the RACF service machine's 305 disk.

If a profile is created that does not conform to the standard format, it is the responsibility of the RACHECK preprocessing exit routine to ensure that the RACHECK SVC does not refer to that profile (that is, does not return a code of 0 to RACHECK when the PROFILE option is specified).

Postprocessing Exit (ICHRCX02)

The 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:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24

When the RACHECK postprocessing exit routines receive control, RACF has already performed the main function (for example, authorization checking), but has not performed any logging or statistics recording.

The ICHRCXP macro maps the RACHECK exit parameter list. To find a mapping of RCXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the RACHECK Postprocessing Exit

When the RACHECK 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 RACHECK SVC, the meanings of which are documented in *z/VM: Security Server RACROUTE Macro Reference*.

Code
Meaning

0
Continue with RACHECK processing. (If the exit routine changes the return- or abend-code values, RACHECK will use these codes.)

4
Try the RACHECK SVC again; invoke the RACHECK 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, RACHECK issues an abend code (382) that indicates a nonvalid exit return code.

Modifying the ICHRCX02 Exit

If you want to modify ICHRCX02 ASSEMBLE and ICHRCX02 TEXT, follow the instructions in "Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List" on page 161, using the following substitution values:

- For *fn* use **ICHRCX02**
- For *nnnn* use **0002**

Deleting the ICHRCX02 Exit

If you want to delete ICHRCX02, you need to perform local modifications to the RPIBLLPA build list. Follow the instructions in "Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List" on page 161, using the following substitution values:

- For *fn* use **ICHRCX02**
- For *blist* use **RPIBLLPA**
- For *nnnn* use **0002**

Note:

1. This process comments out the ICHCRX02 TEXT from the RACFLPA LOADLIB only.
2. Because you are not assembling a file, you can skip that step.

To verify that the exit has been removed, look at the ICH508I message that is displayed during RACF initialization. ICHRCX02 should not be an active exit.

Adding the ICHRCX02 Exit

To add ICHRCX02 TEXT, perform modification to the RPIBLLPA build list. Follow the instructions in "Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List" on page 161 using the following substitution values:

- For *fn* use **ICHRCX02**
- For *blist* use **RPIBLLPA**
- For *nnnn* use **0002**

Note:

1. This process uncomments the ICHRCX02 TEXT section in the RACFLPA build list.
2. If you are simply enabling the sample ICHRCX02 TEXT shipped with the product, then you are not assembling a file and can skip those steps.

RACROUTE REQUEST=DEFINE Exits

The term RACDEF originally referred to the independent system macro. The RACDEF macro was replaced by RACROUTE REQUEST=DEFINE. We recommend that customers use RACROUTE requests in their applications. Many customers are familiar with the term RACDEF, and we continue its use in the RACF library.

The purpose of the RACDEF request 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.

Many of the values passed to the RACDEF preprocessing and postprocessing exits are derived from the parameters specified on the RACROUTE REQUEST=DEFINE macro instruction. For details on this macro instruction, see *z/VM: Security Server RACROUTE Macro Reference*.

Preprocessing Exit (ICHRDX01)

The preprocessing exit routine must be named ICHRDX01.

The exit must be reentrant and is invoked in supervisor state, with protection key 0, with no locks held.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library

- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHRDX01**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRDX01**

When the RACDEF preprocessing exits receive control for the DATASET class, RACF has already processed the naming conventions table, if there is one.

The ICHRDXP macro maps the RACDEF exit parameter list. To find a mapping of RDXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the RACDEF Preprocessing Exit

When the RACDEF 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/VM: Security Server RACROUTE Macro Reference*.

Hex	(Decimal)	Meaning
0	(0)	Exit-routine processing is complete. Normal processing is to continue.
4	(4)	The request is not accepted and is to be failed.
8	(8)	The request is accepted. No more processing is to be performed.
C	(12)	The request is accepted. Processing continues, but authorization checking is bypassed.

Note:

1. If register 15 contains any other value other than those in the table on this page. RACDEF issues a completion code (385) that indicates a nonvalid exit return code.
2. A return code of 4 from the preprocessing exit for an ADDVOL request results in abend 385-4 (nonvalid return code).
3. The RACDEF 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 RACDEF preprocessing exit, see the description of the naming convention exit, ICHCNX00. The ICHCNXP macro maps the naming convention parameter list. To find a mapping of CNXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Postprocessing Exit (ICHRDX02)

The 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:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHRDX02**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRDX02**

The ICHRDXP macro maps the RACDEF exit parameter list. To find a mapping of RDXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the RACDEF Postprocessing Exit

When the RACDEF 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/VM: Security Server RACROUTE Macro Reference*.

Code
Meaning

0
Exit-routine processing is complete. Normal processing is to continue.

4
Retry the RACDEF function; invoke the RACDEF 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, RACDEF issues a completion code (385) that indicates a nonvalid exit return code.

RACROUTE REQUEST=FASTAUTH Exits

The term FRACHECK originally referred to the independent system macro. The FRACHECK macro was replaced by RACROUTE REQUEST=FASTAUTH. We recommend that customers use RACROUTE requests in their applications. Many customers are familiar with the term FRACHECK, and we continue its use in the RACF library.

FRACHECK examines the auditing and global options in effect for the resource while it determines the access authority of the caller. FRACHECK returns a reason code that indicates whether RACHECK should be invoked to log the access attempt. The FRACHECK exits allows the installation to make additional security checks or to instruct FRACHECK to either accept or fail a request.

Note: The FRACHECK exits do not get control during authorization requests for the PROGRAM class and may not be used to affect PROGRAM processing.

Preprocessing Exit (ICHRFX01)

The FRACHECK preprocessing exit must be named ICHRFX01. It is entered before the FRACHECK service routine performs authorization checking.

This exit must be reentrant.

RACF for z/VM does not use the FRACHECK macro in processing CP requests; however, installation-provided RACF exit routines (or other installation-supplied programs that run in the RACFVM service machine) may issue the FRACHECK macro.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 14.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see “Build or Link-Edit a Library” on page 166, using the following substitution values:

- For *fn* use **ICHRFX01**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRFX01**

When you write the FRACHECK exit routine, it is extremely important to be aware of the environment in which the routine executes. This routine is *not* invoked using standard linkage conventions. Its running environment offers limited function as indicated in the following list:

- The execution key is unpredictable.
- The exit may receive control in either supervisor or problem state.
- The exit may or may not be given control APF-authorized.
- The exit should not issue any SVCs.
- The exit routine may be given control in either 24- or 31-bit mode.
- The exit is responsible for saving and restoring certain registers it uses. The FRACHECK exit parameter list contains a pointer (RFXWA) to a 16-byte 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.
  – R14 contains the return address.
  – R15 contains the address of the exit entry point.

  If the exit changes any of the registers (R5 through R13), the exit must save those registers before changing them, and restore them before returning to RACF. The exit may modify any of R0 through R4, R14, and R15 without restoring the value the register had on entry to the exit.

  Of course the R14 value is needed to return to RACF.

The ICHRFXP macro maps the FRACHECK exit parameter list. To find a mapping of RFXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the FRACHECK Preprocessing Exit

On return from the exit routine, FRACHECK checks register 15 for one of the following codes:

Code
Meaning

0
FRACHECK is to continue processing the request.

4
FRACHECK is to fail the request. FRACHECK will return to its caller with a return code of 8.

8
FRACHECK is to accept the request. No further processing is performed by FRACHECK, and FRACHECK's caller receives control with a return code of 0.

Any other code from the exit is treated as an error, and FRACHECK returns to its caller with a return code of 10 (hexadecimal).

If the exit returns with R15=4 or 8, the exit is responsible for setting the 12th word of the work area that RFXWA points to, as follows:

- 0 if the exit does not wish the request to be audited
- 4 if the exit wishes the request to be audited

Also, when returning with R15=4 or 8, the exit should set the 14th word of the work area to 0 or to the address of a profile that the caller can use. This profile must be described by DSECT RACRPE and the DSECTS that follow it.

For more details on this format, see the mapping macro ICHPISP of the ISP data area.

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 exit and RACF's caller.

Postprocessing Exit (ICHRFX02)

The FRACHECK postprocessing exit must be named ICHRFX02. It receives control after the FRACHECK service routine completes processing and before it returns control to the FRACHECK issuer. If there is an FRACHECK preprocessing exit (ICHRFX01) and it sets a nonzero return code, ICHRFX02 is not called.

This exit must be reentrant.

The z/VM system does not use the FRACHECK macro; however, installation-provided RACF exit routines (or other installation-supplied programs that run in the RACFVM service machine) may issue the FRACHECK macro.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 14.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see “Build or Link-Edit a Library” on page 166, using the following substitution values:

- For *fn* use **ICHRFX02**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRFX02**

Return Codes from the FRACHECK Postprocessing Exit

On return from the exit routine, FRACHECK sets one of the following codes in register 0:

Code
Meaning

0
The FRACHECK return code indicates whether the caller is authorized or not to access the resource, and the access attempt is not within the scope of the audit or global-audit specification, making logging unnecessary.

4
The FRACHECK return code indicates whether the caller is authorized or to access the resource, and the access attempt is within the scope of the audit or global-audit specification. The FRACHECK caller should log the attempt by issuing RACHECK for the resource that the caller is attempting to access. This RACHECK will not cause RACF database I/O, because the RACROUTE profiles will be used.

The postprocessing exit routine must return to ICHRFC00 with a return code of 0. ICHRFC00 treats any other return code as an error, and returns to its caller with a return code of 10 (hexadecimal). The 16-word work area pointed to by the parameter list will contain the following information:

- Word 12 contains the reason code that ICHRFC00 passes back to the FRACHECK caller in register 0.
- Word 13 contains the return code that ICHRFC00 will pass back to the FRACHECK caller.
- Word 14 contains the address of the profile (discrete or generic) that ICHRFC00 used to determine authorization, or zero if no profile was found.
- Word 15 contains whatever value was stored there by the ICHRFX01 preprocessing exit routine, or zero if there is no preprocessing exit routine.

RACROUTE REQUEST=LIST Exits

The term RACLIST originally referred to the independent system macro. The RACLIST macro was replaced by RACROUTE REQUEST=LIST. We recommend that customers use RACROUTE requests in their applications. Many customers are familiar with the term RACLIST, and we continue its use in the RACF library.

RACLIST is used by products requiring high-performance authorization checking (such as IMS and CICS®). They then use the FRACHECK service, possibly followed by the RACHECK service, to do authorization checking. If you need to create an authorization checking exit for IMS or CICS, you may need to use a FRACHECK exit or both a FRACHECK exit and a RACHECK exit.

ICHRLX01 is entered before RACLIST builds any in-storage profiles of RACF-defined resources and again after the profiles have been built (at the end of RACLIST processing). ICHRLX02 is entered as each profile is being built.

The RACLIST 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 RACLIST 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 RACLIST selection exit can also resolve conflicts for the OWNER field.

Pre- and Postprocessing Exit (ICHRLX01)

The RACLIST 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:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see “Build or Link-Edit a Library” on page 166, using the following substitution values:

- For *fn* use **ICHRLX01**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRLX01**

The ICHRLX1P macro maps the RACLIST exit parameter list. To find a mapping of RLX1P, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from ICHRLX01

On return from the ICHRLX01 exit routine, RACLIST checks register 15 for one of the following return codes:

Code
Meaning

0
RACLIST is to continue processing.

4
RACLIST is to terminate processing. For a return code, RACLIST 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 RACLIST returns to its caller with a return code of 14 (hexadecimal).

Selection Exit (ICHRLX02)

The RACLIST 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:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 0.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHRLX02**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHRLX02**

The ICHRLX2P macro maps the RACLIST selection exit parameter list. To find a mapping of RLX2P, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the RACLIST Selection Exit

On return from the RACLIST selection exit routine, RACLIST checks register 15 for one of the following return codes:

Hex	(Decimal)	Meaning
0	(0)	RACLIST is to continue processing.
4	(4)	RACLIST 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)	RACLIST 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)	RACLIST is to terminate all processing. The return code passed to the exit in the preprocessing exit's list will be used as RACLIST'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 RACLIST returns to its caller with a return code of 14 (hexadecimal).

Data Set Naming Conventions Table (ICHNCV00)

RACF requires a data set name format in which the high-level qualifier of a data set name is a 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 conventions table by using the ICHNCONV macro. (For information on coding the ICHNCONV macro, see *z/VM: RACF Security Server 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 calls the naming convention table-processing routine, ICHNRT00, before it calls the following exit routines:

ICHRDX01, RACDEF preprocessing exit routine
ICHRCX01, RACHECK preprocessing exit routine
ICHCNX00, command naming convention exit routine
ICHCCX00, command naming convention 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.

The exit must:

- Be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Be reentrant
- Have an AMODE of 24
- Be link-edited into the RACFLPA load library

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use ICHNCV00
- For *blist* use RPIBLLPA
- For *memname* use **ICHNCV00**

Command Exits

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 RACHECK macros to check the command user's authority to a specified resource. The RACHECK preprocessing and postprocessing exits therefore gain control from these commands. In addition, the ADDSD and DELDSD commands use the RACDEF macro to accomplish the data set definition, which means that the RACDEF 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 RACDEF preprocessing exit and the RACHECK 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 convention processing routine before ICHCNX00 receives control. See also “Data Set Naming Conventions Table (ICHNCV00)” on page 127.

This exit must be reentrant.

This exit is not normally used on z/VM. The exception is when z/VM shares a RACF database with z/OS, which is supported on only z/VM 7.2 and earlier versions.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 14.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see “Build or Link-Edit a Library” on page 166, using the following substitution values:

- For *fn* use **ICHCNX00**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHCNX00**

The ICHCNXP macro maps the command preprocessing exit parameter list. To find a mapping of CNXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

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 the following table for a summary of these parameters.

ICHCNX00-Exit Parameter Processing

OFFSET													
CALLER		**0**	**4**	**8**	**12**	**16**	**20**	**24**	**28**	**32**	**36**	**40**	**44**
RACHECK		P	P	P	C	0	C	0	P	C	0	0	0
RACDEF	DEFINE	P	P	P	C	0	C	0	P	C	C	0	0
	RENAME	P	P	P	C	C	C	0	P	C	C	0	0
	ADDVOL	P	P	P	C	0	C	C	P	C	0	0	0
	DELETE	P	P	P	C	0	C	0	P	C	0	0	0
ADDSD	SET	P	P	P	C	0	P[3]	0	P	C	C	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	C	0	P
ALTDSD	SET	P	P	P	C	0	P[3]	0	P	C	0	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	0	0	P
DELDSD	SET	P	P	P	C	0	P[3]	0	P	C	C	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	C	0	P
LISTDSD	Prelocate	P	P	P	C[1]	0	P	0	P	0	0	0	P
	DATASET	P	P	P	C	0	P	0	P	C	0	C	P
	ID or PREFIX	P	P	P	C	0	P	0	P	C	0	C	P
PERMIT	TO resource	P	P	P	C	0	P[3]	0	P	C	0	0	P
	FROM resource	P	P	P	C	0	P[4]	0	P	C	0	0	P
SEARCH	Presearch	P	P	P	C[2]	0	0	0	P	0	0	0	P
	Postsearch	P	P	P	C	0	P	0	P	C	0	0	P
IRRUT100	IRRUT100		P	P	P	C	0	0	0	P	C	0	0
RACXTRT		P	P	P[5]	C	0	C[3]	0	P	C	P[5]	0	0

Note:

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 may 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.

Hex	(Decimal)	Meaning
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 may 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 normal RACF command message is 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, RACDEF DEFINE, and RACDEF 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, RACDEF DEFINE, and RACDEF 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:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library
- Is invoked in supervisor state, key 14.

For information on link-editing or rebuilding the RACFLPA load library to include this exit, see "Build or Link-Edit a Library" on page 166, using the following substitution values:

- For *fn* use **ICHCCX00**
- For *blist* use **RPIBLLPA**
- For *memname* use **ICHCCX00**

This exit is not normally used on z/VM; however, if you are sharing a RACF database with z/OS and issuing data set commands from the z/VM side, the data set commands can invoke these exit routines.

The ICHCCXP macro maps the ICHCCX00 exit parameter list. To find a mapping of CCXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

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.

Password-Encryption Exit

When KDFAES Is Not Active

The password encryption exit routine, ICHDEX01, is called by the RACF manager whenever it is necessary to store or compare encrypted password or password phrase data in a user profile. This exit is also called by RACROUTE REQUEST=EXTRACT processing when TYPE=ENCRYPT,ENCRYPT=(...,INST) is specified.

The exit enables an installation to do the following:

- Use its own authentication exit
- Continue using only the masking algorithm to perform encoding; the IBM-supplied exit sets this return code
- Use only the RACF DES algorithm to perform authentication (see "The RACF DES Algorithm" on page 30 for more information).

If an installation wants to use its own method of user verification, it can set the return code in the ICHDEX01 exit to 0. As a result, RACF uses the encoding routine that the installation has coded in the exit.

If an installation wants to continue using the masking algorithm as the only means of logon checking, it can set the return code in the exit to 4.

If an installation wants to use only the RACF DES algorithm for checking user IDs, it can set the return code in the ICHDEX01 exit to 8. This may be the method you want to use if your installation is a new user of RACF and has never used the masking algorithm.

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 Processing

This exit must be reentrant and is invoked in supervisor state, under protection key 0 and with no locks held.

RACF may have enqueued on the RACF database containing the user profile (either a shared or exclusive enqueue) and may have reserved the DASD volume on which it is located. The exit may not issue any RACF macros or call the RACF manager.

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24

This exit is also called by RACROUTE REQUEST=EXTRACT processing when TYPE=ENCRYPT,ENCRYPT=(...,INST) is specified.

The ICHDEXP macro maps the password encryption exit parameter list. To find a mapping of DEXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

Return Codes from the Encryption Exit

Encrypt Operation

For an *encrypt* operation, when the encryption exit routine returns control, register 15 should contain one of the following return codes.

Hex	(Decimal)	Meaning
0	(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	(4)	The exit has not encrypted the data. RACF is to encrypt the data, using the masking algorithm.
8	(8)	The exit has not encrypted the data. RACF is to encrypt the data, using the RACF DES algorithm.
10	(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.

Compare Operation

For a *compare* operation, when the encryption exit routine returns control, register 15 should contain one of the following return codes.

Hex	(Decimal)	Meaning
0	(0)	The clear text data and the encrypted data should be considered equal.
4	(4)	The exit cannot make a determination. RACF is to attempt to compare the data by using the masking algorithm.
8	(8)	The exit cannot make a determination. RACF is to attempt to compare the data by using the RACF DES algorithm.
C	(12)	The clear text data and the encrypted data should be considered unequal.
10	(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.

Modifying the ICHDEX01 Exit

If you want to modify ICHDEX01 TEXT, follow the instructions in "Modify Full-Part Replacement TEXT Files and Build List" on page 164, using the following substitution values to build the RACFOBJ TXTLIB and RACFLPA LOADLIB:

- For *fn* use **ICHDEX01**
- For *nnnn* use **0002**

Deleting the ICHDEX01 Exit

If you want to delete ICHDEX01 TEXT, you need to preform a local modification to the RPIBLLPA build list. Follow the instructions in "Modify Full-Part Replacement TEXT Files and Build List" on page 164, using the following substitution values:

- For *fn* use **ICHDEX01**
- For *blist* use **RPIBLLPA**
- For *nnnn* use **0002**

Note:

1. This process comments out the ICHDEX01 TEXT from the RACFLPA LOADLIB only.
2. Because you are not assembling a file, you can skip those steps.

Adding the ICHDEX01 Exit

To add ICHDEX01 TEXT, perform modification to the RPIBLLPA build list. Follow the instructions in "Modify Full-Part Replacement TEXT Files and Build List" on page 164 using the following substitution values:

- For *fn* use **ICHDEX01**
- For *blist* use **RPIBLLPA**
- For *nnnn* use **0002**

Note:

1. This process uncomments the ICHDEX01 TEXT section in the RACFLPA build list.
2. If you are simply enabling the sample ICHDEX01 TEXT shipped with the product, then you are not assembling a file and can skip those steps.

SMF Records Exit (ICHRSWX1)

ICHRSWX1 is called after SMF records have been written to disk. This exit can then be used to relay SMF records to other guests. Critical SMF records can be dynamically monitored and, if necessary, specific actions can be taken.

ICHRSWX1 does not receive control for suppressed records. No modification or suppression of records is possible.

The exit:

- Must be reentrant
- Must have an AMODE of 24
- Must be link-edited into the RACFLPA load library.

SMF recording is not affected if ICHRSWX1 terminates abnormally.

ICHRSWX1 may provide return code in register 15 before returning control. If the return code is greater than 0, the return code might indicate that a problem was detected during exit processing. ICHRSWX1 must be compliant with the specification of this exit routine environment.

SMF processing passes to ICHRSWX1 the address of a fullword that points to a parameter list.

Registers at Entry

The contents of the registers on entry to the exit are as follows:

```
0      Not applicable
1      Address of the parameter list
2-12   Not applicable
13     Register save area
14     Return address
15     Entry point address of ICHRSWX1
```

Parameter Descriptions

Register 1 points to the following parameter list:

Word 1
The number of the ensuing parameter; for example, 1.

Word 2
The address of the written SMF record.

Registers at Exit

Upon return from the exit processing, the register contents must be as follows:

```
0-14    Restored to contents at entry
15      0 -- no exceptions occurred; exit finished successfully
        Other values -- not applicable
```

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
- On z/OS, 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.

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/VM: RACF Security Server Auditor's Guide*.

ICHRSMFE Processing

The exit:

- Must be able to run in an XA-mode virtual machine that is using the 370 accommodation facility
- Must be reentrant
- Must have an AMODE of 24

The ICHRSMXP macro maps the report writer exit parameter list. To find a mapping of RSMXP, look in the RACF MACLIB file on the RACF service machine's 305 disk.

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.

Modifying the ICHRSMFE Exit

You need to perform local modifications to the ICHRSMFE TEXT file after you have created the ASSEMBLE file. Follow the instructions in "Modify Full-Part Replacement TEXT Files" on page 163, using the following substitution values to build the RACFLINK LOADLIB:

- For *fn* use **ICHRSMFE**
- For *nnnn* use **0002**

SAF Router Exits

The system authorization facility (SAF) and the SAF router are present on all z/OS systems, even if RACF is not installed. Although the SAF router is not part of RACF, many system components and programs will invoke RACF through the RACROUTE macro and SAF. Therefore, installations can modify RACF parameter lists and do customized security processing within the SAF router. On z/VM, the SAF function is implemented within the RACF service machine. Only ICHRTX00 is supported.

For information about the SAF router exits (ICHRTX00), see *z/VM: Security Server RACROUTE Macro Reference*

Chapter 7. Recovery Procedures

Overview

Most activity against the RACF database is read-only, and RACF uses the primary database exclusively. 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, 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 76.

If running IRRUT200 does not identify the problem, you may want to run the RACF database unload utility program, IRRDBU00. This utility can be used to identify the profile in error. For information on IRRDBU00, see *z/VM: RACF Security Server Security Administrator's Guide*.

After running either IRRUT200 or IRRDBU00, first try to use normal RACF commands to fix the error. If that fails, you may need to use the block update utility command (BLKUPD) to modify your RACF database. For more information on BLKUPD, see *z/VM: RACF Security Server 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 142.

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 for data sets, RACF error messages may specify qualifiers supplied by the exits rather than the high-level qualifiers of the data set names.

The RVARY Command

With the RVARY command you can switch, activate, or deactivate RACF databases without an IPL. You can also use RVARY to list the current configuration of RACF databases.

During recovery, you want to keep the primary database active and not go into failsoft processing. For that reason, we recommend that you use RVARY SWITCH rather than RVARY INACTIVE.

You can enter this command from an active z/VM session

If you have multiple RACF databases, you can use the DATASET operand on the RVARY command to specify which one you want switched, deactivated, or reactivated.

Shared Database Considerations

The use of the RVARY command becomes more complex when the RACF database is shared with other systems. As a general rule, all systems must be synchronized with respect to the RACF database configuration.

© Copyright IBM Corp. 1993, 2025

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.

In a multiple service machine environment on z/VM, the RVARY command only inactivates the database on the server that processed the RVARY. For information on directing the RVARY command to multiple RACF service machines, see the RAC command in *z/VM: RACF Security Server Command Language Reference*.

See *z/VM: RACF Security Server Command Language Reference* for the complete syntax of RVARY and considerations when issuing RVARY in a multiple service machine environment.

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 switch the RACF database. 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 completion of the request (switch, activate, or deactivate) to the RACF database.

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. Once the operator receives a password, the security administrator should then change the password for security purposes.

RVARY SWITCH

If you have a backup database specified in the database name table (ICHRDSNT), you can enter the RVARY SWITCH command to switch from the failing primary database to the backup database.

Before entering an RVARY SWITCH, you must ensure that the backup database is active. The SWITCH option of the command deactivates the current primary database and causes RACF to use the backup copy as the new primary. You should repair the old primary and activate it at the earliest opportunity.

When an RVARY SWITCH command is issued, the database buffers are also switched. RACF associates a set of buffers with the new primary database (the old backup database) and disassociates the buffers from the old primary database (the new backup database). The database buffers are switched when the primary and backup database formats are the same and when the databases are both on shared devices or both on non-shared devices.

Attention:
When you enter RVARY SWITCH from your backup database to return to your primary database, your backup database will be automatically deactivated and deallocated; therefore, you must enter the RVARY ACTIVE command to reallocate and reactivate it.

RVARY ACTIVE | INACTIVE

Without a Backup Database

If your installation does not have a backup database specified in the database name table (ICHRDSNT), and you need to deactivate the primary database, you have no choice but to use the RVARY INACTIVE command. If you have a single database, this puts you into failsoft processing. If you have multiple databases and only one is inactive, 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 one active.

Failsoft Processing

Failsoft processing occurs when there are no primary RACF databases available (RACF is installed but inactive). Although it degrades system performance and system security, in rare cases it may be necessary when you repair RACF. During failsoft processing the operator is prompted frequently to grant or deny access to data sets. The resource manager decides on the action for general resource classes with a return code of 4.

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

- RACF is installed but does not know the name of the database.
- Failures occurred during RACF initialization at IPL time.
- An RVARY INACTIVE command was issued (inactivating all primary databases).

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 RACHECK and RACDEF 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.

Failsoft processing is not active if you have deactivated RACF with the SETRACF INACTIVE command. Users can still log on; however, since you have turned off the RACF service machine, RACF processing is not possible.

General Considerations

- A RACF database that is shared by two systems is deactivated only for the system from which you entered the RVARY command. You should deactivate all systems sharing a database, as results can be unpredictable.

  In a multiple service machine environment on z/VM, the RVARY command only inactivates the database on the server that processed the RVARY. For information on issuing the RVARY command in a multiple RACF service machine environment, see the RAC and RVARY commands in *z/VM: RACF Security Server Command Language Reference*.
- If a RACF database is deactivated, the system operator must be aware that there will be many prompts that will have to be responded to.
- If 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.
- The RACHECK and RACDEF 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 RACDEF 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.
- You cannot enter RACF commands to make changes to profiles on a deactivated database.
- If you have more than one primary database, you must enter RVARY INACTIVE for all of your primary databases for failsoft processing to be in effect. If you enter RVARY INACTIVE for only one of the primary databases, failsoft processing will not be in effect; therefore, any RACF activities involving that database will fail.

- You can use exit routines to examine the data set descriptor table created during RACF initialization and determine if a RACF database has been deactivated by the RVARY command.

Impact on Users

Failsoft processing affects users in the following ways:

- **z/VM users already logged on:**

  If RACF becomes inactive after initialization, those users already on the system continue to have their access requests validated by RACF. 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 resource and the decision could not be made using a valid internal table RACF, through failsoft processing, prompts the operator to approve the request.
- **z/VM users not logged on:**

  z/VM users will be unable to log on while the RACF database is inactive.

Synchronization Considerations

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 Database

On z/VM, an EXEC might be useful to analyze discrepancies.

Other Recovery Considerations

You can use RACF commands to accomplish all the synchronization steps (some may require the SPECIAL attribute). Zap and similar programs are unnecessary, and you should not use them. Similarly, use of the BLKUPD utility 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.

Recovery

If there are problems with the primary RACF database and it needs repair, the operator or other designated user can enter RVARY SWITCH from the RACF service machine to bring up the backup database.

However, if there is a problem with the RACF service machine itself that prevents the user from entering RVARY SWITCH, then the operator must log off the RACF service machine and enter an AUTOLOG or XAUTOLOG command for RACMAINT, the backup service machine. The system programmer can then log on to the RACF service machine and fix the problem. See “Failures on the RACF Service Machine” on page 148.

Failures on the RACF Database

In the unlikely event of I/O failures against the device upon which the RACF database resides, or in the case of RACF database corruption, one of the following situations may 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

Sample recovery procedures are provided below for each situation.

If you have split your database and only one database is in error, only the broken database must be recovered. When you issue the RVARY command, name the broken database using the DATASET operand. In general, do not let the database name default. By using the DATASET operand, you also avoid accidentally processing the wrong database.

The Primary Database Is in Error; the Backup Database Is Unaffected

In this situation, follow this procedure:

1. Ensure that the backup is active.
2. Issue RVARY SWITCH (the backup is now the *new primary*).
3. Do one of the following:
   - Correct the problem on the old primary, using BLKUPD.
   - If the device is accessible, copy the backup (*new primary*) onto the old primary, using IRRUT200 or IRRUT400.
   - If the device is inaccessible: allocate, catalog and copy a replacement primary onto a different DASD device, using IRRUT200 or IRRUT400.
4. Issue RVARY ACTIVE for the old primary or replacement primary.
5. Issue RVARY SWITCH.
6. Issue RVARY ACTIVE for the backup (*original backup*).

The Backup Database Is in Error; the Primary Database Is Unaffected

In this situation, follow this procedure:

1. Issue RVARY INACTIVE for the backup.
2. Do one of the following:
   - Correct the problem on the backup, using BLKUPD.
   - If the device is accessible, copy the primary onto the backup, using IRRUT200 or IRRUT400.
   - If the device is inaccessible, allocate, catalog, and copy a replacement backup onto a different DASD device, using IRRUT200 or IRRUT400.
3. Issue RVARY ACTIVE for the backup.

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 141.
2. Obtain the most recent dump of your RACF database.
3. Do one of the following:
   - If the device is accessible, copy the dump to the primary database.
   - If the device is inaccessible, allocate, catalog, and copy the database onto a different DASD device.
4. Issue RVARY ACTIVE.

   Your database is probably back-level. To bring it up to date, use a combination of the SMF records and the RACF report writer to add or delete the appropriate profiles and access authorities.

Both the Primary and the Backup Databases Are in Error

In this situation, follow the procedure for the situation in which your primary database is in error and you have no backup database.

Once you have the primary database, follow the procedure for the situation in which the backup database is in error and the primary database is unaffected.

Failures 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: The operator must have a specific authority to enter some command operands. See *z/VM: RACF Security Server Command Language Reference* for more information about these commands and their operands.

Commands That Do Not Modify RACF Profiles

The commands that do not modify RACF profiles are:

- DISPLAY (RACF operator command)
- LISTDSD
- LISTGRP
- LISTUSER
- RLIST
- RVARY
- SEARCH
- SETROPTS

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 index 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 your programming support representative.

Commands That Have Recovery Routines

Failures that occur during the processing of the following commands may or may 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. See *z/VM: RACF Security Server Security Administrator's Guide* for more information.

   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, alter 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.

   See *z/VM: RACF Security Server Security Administrator's Guide* for information regarding VMPOSIX mapping profiles. For information on the NOADDCREATOR option, see *z/VM: RACF Security Server Security Administrator's Guide*. For information on the ADDCREATOR and NOADDCREATOR keywords on the SETROPTS command, see *z/VM: RACF Security Server Command Language Reference*.
6. If there are no discrepencies and the user, group, and 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.

The commands are:

- ADDGROUP
- ADDSD
- ADDUSER
- ALTDSD (with the ADDVOL, ALTVOL, or DELVOL operand)
- ALTGROUP
- ALTUSER
- DELDSD
- DELGROUP
- DELUSER
- 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, data set, and VMPOSIX mapping profiles to determine the status of the contents.
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. See *z/VM: RACF Security Server Security Administrator's Guide* for more information on VMPOSIX mapping 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. See *z/VM: RACF Security Server Security Administrator's Guide* for more information on VMPOSIX mapping profiles. For information on the NOADDCREATOR option, see *z/VM: RACF Security Server Security Administrator's Guide*. For information on the ADDCREATOR and NOADDCREATOR keywords on the SETROPTS command, see *z/VM: RACF Security Server Command Language Reference*.

5. If the failure occurs again, contact your programming support representative.

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.

For messages IRR402I, IRR403I, and IRR404I, see *z/VM: RACF Security Server 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(s) or profile entry(s) in your RACF database. Since the index structure is required to locate profile data, it is essential to have a valid index structure. Therefore the following steps should be performed in order during problem determination to find the failing profile.

   a. Run the RACF database verification utility program (IRRUT200) 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 5, “Utilities for the RACF Database,” on page 67.

      If IRRUT200 does not detect any problems in the RACF database structure, (it verifies the index structure down to the profile level), you may try running the RACF Database Unload utility, (IRRDBU00). The IRRDBU00 utility must read every profile in the database and thereby (implicitly) may identify profiles with errors. If IRRDBU00 encounters a profile in error, it may 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 may also be able to determine the profile in error. To do this, look in the output dataset (OUTDD) and find the last profile, (at the bottom), that was unloaded. It is likely that this profile is okay, however; the next profile in the database, (in the same class), is likely to be the culprit if indeed a bad profile is causing the utility to terminate.

   b. Attempt to correct the problem by using normal RACF commands. If this does not work, use the block update (BLKUPD) utility command 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 the BLKUPD utility command to correct the additional problems.

For messages IRR402I, IRR403I, and IRR404I, the system programmer or security administrator should perform steps 2a and 2b.

Failures on the RACF Service Machine

RACF runs as RACFVM in a disconnected virtual service machine. If RACFVM fails, you can activate RACMAINT as a backup RACF service machine. RACMAINT uses backup minidisks, which are created when you install and service RACF, as its primary minidisks until you repair RACFVM. This arrangement allows you to switch between the backup and production user IDs. For a detailed description of the RACF service machines, see <*RACF Program Directory*.

If the primary RACF service machine (RACFVM) fails, the primary system operator can resume RACF service in the following way:

- Ensure that the RACMAINT user ID is set up as described in *RACF Program Directory*.
- Enter the FORCE command to force off RACFVM.
- Enter the AUTOLOG or XAUTOLOG command to automatically log on RACMAINT.

After you have determined and corrected the cause of RACFVM's failure, resume normal operation by having the primary system operator switch back to the RACFVM service machine by following the steps listed below:

- Enter the FORCE command to force off RACMAINT.
- Enter the AUTOLOG or XAUTOLOG command to automatically log on RACFVM.

When the RACF Service Machine Is Uninitialized or Unresponsive

When the RACF service machine is uninitialized or unresponsive, no users are allowed to log on to the system, except for the following:

- Any RACF service machine
- The primary system operator or any of the defined alternate system operators, if there is no system operator currently logged on.
- The userid currently logged on as system operator when the HERE operand is used on the LOGON command.

For these user IDs, the request is deferred to CP which checks the password supplied by the user against the password in the user's entry in the CP directory.

Note: By coding LOGONBY directory statements for the system operator user IDs or for RACF service machines, you still allow the use of LOGON BY for these user IDs when the RACF service machine is experiencing problems. This could be useful for recovery purposes.

Chapter 8. Storage Estimates

This section provides information for estimating the RACF storage requirements for:

- RACF databases
- System libraries
- Interactive System Productivity Facility (ISPF) data files
- RACF for z/VM servers.

RACF Database Storage Requirements

This section describes two methods for estimating the size of a RACF database. The first method is an approximation; the second is a detailed formula. Note that when a RACF database becomes full, you can extend it with the Split/Merge/Extend utility program (IRRUT400).

Approximation of the RACF Database Size

The direct-access space needed for a RACF database depends mainly on the number of users, groups, user-group connections, and resource profiles defined to RACF. The size also depends on the name lengths of the entities defined to RACF, how efficiently the space within the RACF database is utilized, and the password encryption algorithm in effect.

On the average, a RACF database requires approximately 320,000 bytes for each 1000 entities defined to RACF.

System Library Storage Requirements

Table 8 on page 149 provides approximate storage requirements for z/VM system libraries.

Table 8. Approximate Space Requirements for the z/VM System Libraries for RACF 5.3

Library File	4K BLOCKS	Contents
RACFLINK LOADLIB	375	RACF database-initialization program, RACF report writer, RACF class-descriptor table, and RACF utilities.
RACFCMDS LOADLIB	526	RACF commands.
RACFLPA LOADLIB	244	RACF manager routines, RACF SVC-processing routines, RACF service routines, and installation-written exit routines.
RACFINTF LOADLIB	19	RACF interface logic for z/OS-based code to run in a z/VM environment.
RACF MACLIB	593	RACF template-definition member (ICHTEMP10), and the macros used by or provided by RACF. See *z/VM: RACF Security Server Macros and Interfaces* for a list of these macros.
SYS1 HELP	540	Help messages for RACF command and report writer users.

Note: Add the required blocks for any installation-written exit routines installed for RACF.

Unless you manually change the job stream created by the RACF installation process, the RACF modules reside in RACFLINK LOADLIB, RACFINTF LOADLIB, RACFLPA LOADLIB, and RACFCMDS LOADLIB. If you want the RACF modules to reside in other libraries, remember that the RACF manager, RACF SVC-

© Copyright IBM Corp. 1993, 2025

processing routines, and exit routines run in key 0 and supervisor state, and *must* reside in RACFLPA LOADLIB.

Interactive System Productivity Facility (ISPF) Storage Requirements

Space for the ISPF data files for RACF is obtained by adding 2,494 4KB blocks to the ISPVM's 192 minidisk.

RACF Virtual Storage Requirements

Make sure that you have enough virtual storage for the RACF service machine; 32MB is recommended.

Appendix A. Setting Up Multiple RACF Service Machines

This appendix shows you how to set up multiple RACF service machines. The following sections describe the major tasks you need to perform to use multiple RACF service machines:

- Install the multiple RACF service machines capability
- Initialize the multiple RACF service machines.

Installing Multiple RACF Service Machines

It is recommended that you first install RACF without support for multiple RACF service machines. Then determine whether you want to install additional RACF service machines. This decision should be made by system programmers familiar with the performance characteristics and needs of your system.

Note: Your database cannot reside on FBA DASD if you want to use multiple RACF service machines.

When you are ready to install multiple RACF service machines support, perform the following steps:

1. Each RACF service machine must be defined in the CP directory, as well as in RACF user profiles.

   For information on how to define a RACF service machine in the CP directory, refer to "CP Directory Considerations for Multiple RACF Service Machines" on page 152.

   For each service machine, issue the ADDUSER command using options similar to the original RACFVM user ID options. (To determine RACFVM user ID options issue a LISTUSER for RACFVM.)
2. DASD must be allocated for each additional service machine.

   Each service machine will require a 191, 301, and 302 minidisk. The requirements for cylinder sizes are no different than for the single RACFVM service machine in RACF. For information on the cylinder sizes, refer to *RACF Program Directory*.
3. The PROFILE EXEC of the AUTOLOG1 user ID must be modified to XAUTOLOG the additional RACF service machines.

   Note: Do not alter the AUTOLOG1 user ID PROFILE EXEC until you are ready to activate the multiple RACF service machine capability.

   (The user ID of AUTOLOG1 can be tailored. If you change the name of AUTOLOG1, you need to make the relevant RACF changes to that user ID.)
4. If you want to use the RACF-supplied RACFSMF user ID (the user ID for archiving SMF data), you need to copy the file SMFPROF EXEC (residing on the 305 disk) to the RACFSMF 191 disk with a filename of PROFILE EXEC.

   If the file already exists on the RACFSMF 191 disk, replace it with the version on the 305 disk.

   If you have an application that XAUTOLOGs RACFSMF (to perform archiving), you must change the XAUTOLOG statement to supply the user IDs of all of the service machines as console input data.

   For example, if the RACF service machine user IDs are RACFVM, RACFVM1, and RACFVM2, an XAUTOLOG statement would appear as:

   ```
   XAUTOLOG RACFSMF #RACFVM RACFVM1 RACFVM2
   ```

   Note: The symbol # has special restrictions. For example, it can appear as a logical line end, thereby causing the system to attempt to execute RACFVM (in the example) as a command. For additional information on these restrictions, refer to *z/VM: CP Commands and Utilities Reference*.

   For additional information on RACFSMF and SMF recording, refer to *z/VM: RACF Security Server Auditor's Guide*.

© Copyright IBM Corp. 1993, 2025

5. Change the CSTCONS ASSEMBLE file to use the MESSAGE command (rather than MSGNOH) to issue security messages.

   As a result of this change, the user ID of the originating service machine will be displayed as part of the corresponding message and helps identify which service machine the message comes from.

   For information on updating CSTCONS, see "Message Support" on page 54.
6. Code RACSERV macros to immediately follow the HCPRWATB entry definition label in HCPRWA.

   Code one RACSERV macro for each RACF service machine you need to define. You can define a maximum of 10 RACF service machines.

   For information on the RACSERV macro and for instructions on updating HCPRWA, see *z/VM: RACF Security Server Macros and Interfaces*.

   Attention:

   Do not remove the following instruction. It appears in HCPRWA directly after the RACSERV macros and marks the end of the list of multiple RACF service machines being defined.

   ```
            DC    X'FFFFFFFF'
   ```

   If this statement is moved or deleted, unpredictable behavior will result when the service machine list is scanned during normal system operation.
7. Run the RACFSVRS EXEC.

   RACFSVRS will format the DASD for the additional service machines and copy any necessary files to the newly formatted disks. Refer to "Initializing Multiple RACF Service Machines (RACFSVRS EXEC)" on page 154 for details on the RACFSVRS EXEC.

CP Directory Considerations for Multiple RACF Service Machines

This section describes how to set up the CP directory for additional RACF service machines. The term "master" service machine is used to designate the single service machine that owns DASD shared between all of the service machines. (The term "master" is not used to imply that this service machine is in any way controlling the distribution of work among the other service machines. You can think of the traditional RACFVM user ID as the "master" in relation to the additional service machines. For all practical purposes, any of the service machines can be designated as "master."

Each new service machine's directory entry is essentially a copy of the directory entry that exists in RACFVM with the following exceptions:

- The primary and secondary RACF database minidisks must be defined for the master service machine with the MWV CP attribute of the MDISK statement (which uses CP's virtual reserve/release support). All other service machines must link to the database disks of the master service machine in MW mode.
- The 305 and 490 minidisks must be coded with an MDISK statement in the master service machine, and LINK statements for these disks should exist in the directory entries for the other RACF service machines.

  Note: For the other service machines, READ access is sufficient.

After updating and redirecting the CP directory, make sure to log off the "master" RACF service machine and then log on to it again; this ensures that the RACF database is in synch with your changes.

The following two figures are examples of CP directory entries for RACFVM (the master service machine user ID) and RACFVM*a* (an additional service machine).

```
USER RACFVM  SYS1     32M 32M ABCDEGH
    IUCV *RPI PRIORITY MSGLIMIT 255
    IUCV  ANY PRIORITY MSGLIMIT 100
     ACCOUNT SYSTEMS
     MACH XA
     IPL 490 PARM AUTOCR
     OPTION MAXCONN 300
```

```
     CONSOLE 009 3215 T OPERATOR
     SPOOL 00C 2540 READER *
     SPOOL 00D 2540 PUNCH B
     SPOOL 00E 1403
     LINK MAINT 190 190 RR
     LINK MAINT 19D 19D RR
     LINK MAINT 19E 19E RR
* RACFVM A-DISK
     MDISK 191 3390 xxx 10 yyyyy1 MW SECRETRD SECRETW
* RACFVM B-DISK (system files disk)
     MDISK 305 3390 xxx 080 yyyyy1 MR  SECRETRD SECRETW
* RACFVM SYSTEM DISK
     MDISK 490 3390 xxx 46 yyyyy1 MR  SECRETRD SECRETW
*
* RACFVM A-DISK (backup/recovery)
     LINK RACMAINT 191 591 *
* RACFVM B-DISK (backup/recovery)
     LINK 7VMRAC30 505 505 MR
* RACFVM SYSTEM DISK (backup/recovery)
     LINK 7VMRAC30 590 590 MR
*
* RACF DATA BASE MINIDISK (Performance Sensitive)
     MDISK 200 3390 xxx 020 yyyyy1 MWV SECRETRD SECRETW
* RACF BACKUP DATA BASE MINIDISK (Performance Sensitive -
*                       not on the same drive as PRIMARY)
     MDISK 300 3390 xxx 020 yyyyy2 MWV SECRETRD SECRETW
*
* PRIMARY SMF DATA FILE  (Performance Sensitive)
     MDISK 301 3390 xxx 008 yyyyy3 MR  SECRETRD SECRETW
* SECONDARY SMF DATA FILE  (Performance Sensitive)
     MDISK 302 3390 xxx 008 yyyyy4 MR  SECRETRD SECRETW
*
* where xxx is the starting cylinder number
* and yyyyyN is the volume label, N indicating different packs
```

```
 USER RACFVMa SYS1     32M 32M ABCDEGH  1
     IUCV *RPI PRIORITY MSGLIMIT 255
     IUCV  ANY PRIORITY MSGLIMIT 100
      ACCOUNT SYSTEMS
      MACH XA
      IPL 490 PARM AUTOCR
      OPTION MAXCONN 300
      CONSOLE 009 3215 T OPERATOR
      SPOOL 00C 2540 READER *
      SPOOL 00D 2540 PUNCH B
      SPOOL 00E 1403
      LINK MAINT 190 190 RR
      LINK MAINT 19E 19E RR
 * RACFVMn A-DISK
      MDISK 191 3390 xxx 10 zzzzz1 MW SECRETRD SECRETW
 * RACFVM B-DISK (shared)
      LINK RACFVM 305 305 RR
 * RACFVM SYSTEM DISK (shared)
      LINK RACFVM 490 490 RR
 *
 * RACF DATA BASE MINIDISK (Performance Sensitive, shared)
      LINK RACFVM 200 200 MW
 * RACF BACKUP DATA BASE MINIDISK (Performance Sensitive -
 *                       not on the same drive as PRIMARY, shared)
      LINK RACFVM 300 300 MW
 *
 * PRIMARY SMF DATA FILE  (Performance Sensitive)
      MDISK 301 3390 xxx 008 zzzzz5 MW  SECRETRD SECRETW
 * SECONDARY SMF DATA FILE  (Performance Sensitive)
      MDISK 302 3390 xxx 008 zzzzz6 MW  SECRETRD SECRETW
 *
 * where xxx is the starting cylinder number
 * and zzzzzN is the volume label, N indicating different packs.
```

Note:

- As more servers are added, their 301 and 302 disks should continue to be placed on different packs than any of the previously defined performance-sensitive disks for all servers.
- In the second example, RACFVM*a* is the user ID of the service machine; *a* could represent the numbers 1 through 9.

Initializing Multiple RACF Service Machines (RACFSVRS EXEC)

Use the RACFSVRS EXEC to initialize multiple RACF service machines. The EXEC verifies that each user ID contains required, predetermined properties (for example, correct access to the system disks that support the service machine). If the user IDs are not correctly set up, you receive warning messages. When all of your input is verified, the EXEC formats and initializes those disks that belong to a RACF service machine.

The RACFSVRS EXEC performs the following processing steps:

1. Displays the RACFSVRS screen on your terminal.
2. Verifies input fields. If errors are found, messages are displayed and you are given the choice either to exit the EXEC and correct the problem or to continue (if the error is not severe).
   - For all user IDs, RACFSVRS verifies that the proper links can be made to the required disks by the executing user ID.

     You will be notified of any unexpected conditions and RACFSVRS will only proceed after all required links are verified.

     Links are verified as follows:

     – The master service machine 191 and 305 disks must be READ accessible.
     – Each additional service machine's 191, 301, and 302 disks must be WRITE accessible.
   - RACFSVRS checks each additional service machine disk to make sure that it has not already been initialized:
     – Check the 191 disk for the existence of a PROFILE EXEC and an SMF CONTROL file.
     – Check the 301 and 302 for the existence of an SMF DATA file.
3. Formats each of the additional service machine 191 disks.
4. Places a copy of the PROFILE SAMPLE on each 191 disk as a file named PROFILE EXEC.
5. Places a copy of the SMF CONTROL file on each service machine's 191 disk with the corresponding SMF archiving user ID updated.
6. CMS formats the 301 and 302 SMF recording disks.
7. Finally, makes an entry in the RACFSVRS-created HISTORY file to record the initialized service machine user IDs with a time and date stamp.

Appendix B. Description of the RACF Classes

IBM-Supplied Resource Classes that Apply to z/VM Systems

For a complete listing of all IBM-supplied resource classes in the CDT, see *z/VM: RACF Security Server Macros and Interfaces*.

Class	Purpose
DIRACC	Controls auditing (via SETROPTS LOGOPTIONS) for access checks for read/write access to HFS directories. Profiles are not allowed in this class.
DIRECTRY	Protection of shared file system (SFS) directories.
DIRSRCH	Controls auditing (via SETROPTS LOGOPTIONS) of HFS directory searches. Profiles are not allowed in this class.
FACILITY	Miscellaneous uses. Profiles are defined in this class so resource managers (typically program products or components) can check a user's access to the profiles when the users take some action. Examples are using combinations of options for tape mounts, and use of the RACROUTE interface. RACF does not document all of the resources used in the FACILITY class by other products. For information on the FACILITY-class resources used by a specific product (other than RACF itself), see that product's documentation.
FIELD	Fields in RACF profiles (field-level access checking).
FILE	Protection of shared file system (SFS) files.
FSOBJ	Controls auditing (via SETROPTS LOGOPTIONS) for all access checks for HFS objects except directory searches. Controls auditing (via SETROPTS AUDIT) of creation and deletion of HFS objects. Profiles are not allowed in this class.
FSSEC	Controls auditing (via SETROPTS LOGOPTIONS) for changes to the security data (FSP) for HFS objects. Profiles are not allowed in this class.
GLOBAL	Global access checking. [1]
GMBR	Member class for GLOBAL class (not for use on RACF commands).
GTERMINL	Terminals with IDs that do not fit into generic profile naming conventions. [1]
PROCESS	Controls auditing (via SETROPTS LOGOPTIONS) of changes to UIDs and GIDs of OpenExtensions VM processes. Controls auditing (via SETROPTS AUDIT) of dubbing and undubbing of OpenExtensions VM processes. Profiles are not allowed in this class.
PSFMPL	When class is active, PSF/VM performs separator and data page labeling as well as auditing.
PTKTDATA	PassTicket Key Class.
PTKTVAL	Used by NetView/Access Services Secured Single Signon to store information needed when generating a PassTicket.
RACFEVNT	RACFEVENT class contains profiles which control whether RACF change log notification is performed for USER profiles, and whether password or password phrase enveloping is to be performed.
RACFVARS	RACF variables. In this class, profile names, which start with & (ampersand), act as RACF variables that can be specified in profile names in other RACF general resource classes.

© Copyright IBM Corp. 1993, 2025

Class	Purpose
RVARSMBR	Member class for RACFVARS (not for use on RACF commands).
SCDMBR	Member class for SECDATA class (not for use on RACF commands).
SECDATA	Security classification of users and data (security levels and security categories). [1]
SECLABEL	If security labels are used and, if so, their definitions. [2]
SFSCMD	Controls the use of shared file system (SFS) administrator and operator commands.
SURROGAT	If surrogate submission is allowed, and if allowed, which user IDs can act as surrogates.
TAPEVOL	Tape volumes.
TERMINAL	Terminals (TSO or z/VM). See also GTERMINL class.
VMBATCH	Alternate user IDs.
VMCMD	CP commands, DIAGNOSE instructions, and system events.
VMDEV	Control who connects to real devices.
VMLAN	Use RACF to control Guest LANs
VMMAC	Used in conjunction with the SECLABEL class to provide security label authorization for some z/VM events. Profiles are not allowed in this class.
VMMDISK	z/VM minidisks.
VMNODE	RSCS nodes.
VMRDR	z/VM unit record devices (virtual reader, virtual printer, and virtual punch).
VMSEGMT	Restricted segments, which can be named saved segments (NSS) and discontiguous saved segments (DCSS).
VXMBR	Member class for VMXEVENT class (not for use on RACF commands).
VMXEVENT	Auditing and controlling security-related events (called z/VM events) on z/VM systems.
VMPOSIX	Contains profiles used by OpenExtensions z/VM.
WRITER	z/VM print devices.

Note:

1. You cannot specify this class name on the GENCMD, GENERIC, and GLOBAL/NOGLOBAL operands of the SETROPTS command.
2. You cannot specify this class name on the GLOBAL operand of the SETROPTS command or, if you do, the GLOBAL checking is not performed.

Appendix C. Selecting Options with ICHSECOP

The ICHSECOP module enables you to select the number of resident data blocks (when you don't have a database name table). It enables you to bypass RACF-initialization processing (and RACF is inactive) and lets you disallow duplicate names for discrete data set profiles.

This section describes the following options that you can specify in the ICHSECOP module:

- Bypassing RACF-initialization processing during IPL.
- Selecting the number of resident data blocks (only if there is no data set name table).

  Note: The preferred method of controlling the number of resident data blocks is to use the database name table. If you have specified the number of resident data blocks in both the ICHSECOP module and the database name table, RACF uses the figure in the database name table.
- Disallowing duplicate names for data set profiles.

The RACF program product contains a module (ICHSECOP) that you must replace in order to use these options. When you receive the module from IBM, it is set so that RACF-initialization processing is performed during IPL (and RACF is activated), ten data blocks are made resident, and duplicate data set profile names are allowed.

The module is used only during IPL. When you change the module, the changes are not effective until after the next IPL.

Module ICHSECOP contains 5 bytes of data, formatted as follows:

Table 9. ICHSECOP Module

Bytes	Bits	
0	0 1 2-7	Bypass RACF-initialization processing (when set on) Disallow duplicate names for data set profiles (when set on) Reserved
1-4		The number of data blocks to be made resident

Bypassing RACF-Initialization Processing (on z/OS)

CAUTION: This option is intended for use on z/OS. If you use this option on z/VM, no one can log on or access any resources. If you wish to logically remove RACF from the system, use the SETRACF INACTIVE command instead. See *z/VM: RACF Security Server Command Language Reference* for details.

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 reverts to UADS user identification and verification. Also, RACF commands cannot be issued.

If a user accesses a RACF-protected resource, the RACROUT REQUEST=OFF 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. 1993, 2025

- Instruct the operations staff about the RACF failsoft messages and intervention requests.

The RACDEF SVC 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 RACDEF macro instruction, the failsoft processing in RACDEF will gain control and issue messages to the system operator. The RACDEF failsoft processing also handles a job that has the PROTECT parameter specified on a DD statement. Note that RACDEF 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 database name table (ICHRDSNT). You can use ICHRDSNT to specify the number of resident data blocks for each primary RACF database. (See “Database Name Table” in Chapter 3, “RACF Customization,” on page 19.)

If your installation does not have a database name table, you can specify the number of resident data blocks for a single RACF database in ICHSECOP, or use the default value of 10 resident data blocks. However, be aware that using ICHRDSNT provides more flexibility and better performance options than using ICHSECOP.

If you have a database name table, and have additionally specified resident data blocks using ICHSECOP, the database 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 100. The blocks reside in the RACF service machine's virtual storage.

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.

Appendix D. Using VMSES/E Support for Installation and Customization

During customization of RACF you might have created your own ASSEMBLE or TEXT files, created your own exits, or modified source files. This appendix provides instructions for performing the local modifications needed to use the new or changed files. These instructions include:

- Assembling files
- Modifying files and build lists
- Building a library
- Link-editing a library

VMSES/E Information

For more information on VMSES/E local modifications, see *z/VM: Service Guide* and *z/VM: VMSES/E Introduction and Reference*.

Assemble a File, Modify a Build List, and Build a Library

During customization of RACF you might have created your own ASSEMBLE file. You need to assemble this file and build the text into a RACF library (for example, LOADLIB). You also have to make a local modification to an existing VMSES/E RACF build list to include the new information.

The following instructions explain how to:

- Assemble the file
- Perform the local modification to the build list
- Build the appropriate library

Use the substitution command values provided in the instructions that pointed you to this topic.

1. Assemble the file.

   ```
   VMFASM fn SERVP2P compname (OUTMODE LOCALSAM
   ```

   For *compname*, use:

   RACF
   For installing on minidisks

   RACFSFS
   For installing in shared file system (SFS) directories

   RACFPANL
   For installing on minidisks with RACF ISPF panels

   RACFPANLSFS
   For installing in SFS directories with RACF ISPF panels

   Notes:

   a. Other options are available for the VMFASM command. See *z/VM: VMSES/E Introduction and Reference* for additional information.

   b. If the assemble function is successful, the file *fn* TXT00000 is placed on the local modifications disk (LOCALSAM 2C2).

2. Create and apply a local modification to the build list used to build the library that contains the file.

© Copyright IBM Corp. 1993, 2025

```
LOCALMOD compname blist EXEC (MODID Lnnnn
```

a. Reply **1** to the prompt message.

b. In the displayed file, change the following statement for :OBJNAME *memname*.

```
*:OPTIONS. ORDER name-of-new-classes
```

i) Remove the asterisk (*) in the first column.

ii) Change *name-of-new-classes* to the name of your new class or classes. For more than one class, specify ORDER `class1, class2`, and so forth. This ensures that the class descriptor table links correctly.

Note: Be sure that your linkage editor ORDER statements specify *fn* as the last CSECT. Unpredictable results occur if you omit an ORDER statement or if *fn* is not the last CSECT.

iii) Enter **file** on the XEDIT command line to file your changes.

```
====>file
```

3. Build your new local modification into the library on the test build disk.

```
SERVICE compname BUILD
```

4. Put the new LOADLIB into production.

```
PUT2PROD compname
```

Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Modify Full-Part ASSEMBLE and TEXT Files

During customization of RACF you might have to modify ASSEMBLE files and corresponding TEXT files that are serviced as full-part replacement.

The following instructions describe how to create and build the new parts. Use the substitution command values provided in the instructions that pointed you to this topic.

1. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

For *compname*, use:

RACF
For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

2. If you want to use the New-Password-Phrase Exit (ICHPWX11), continue with step 3. Otherwise, continue with step 6.
3. If you want to use the default IBM-supplied ICHPWX11 TEXT file already assembled, complete steps 4 and 5, skip step 6, and continue with step 7. Otherwise, to modify ICHPWX11 ASSEMBLE, continue with step 6.
4. Copy the ICHPWX11 ASSEMBLE file to the local modifications disk (LOCALSAM 2C2).

```
VMFREPL ICHPWX11 ASSEMBLE SERVP2P compname ($SEL LOGMOD Lnnnn OUTM LOCALSAM
```

5. Copy the ICHPWX11 TEXT file to the local modifications disk (LOCALSAM 2C2).

```
VMFREPL ICHPWX11 TEXT SERVP2P compname ($SEL LOGMOD Lnnnn OUTM LOCALSAM
```

6. Create and apply a local modification to the ASSEMBLE file.

```
LOCALMOD compname fn ASSEMBLE (MODID Lnnnn
```

   a. Reply **1** to the prompt message.
   b. In the displayed file, make your local modification.
   c. Enter **file** on the XEDIT command line to file your changes.

```
====>file
```

   Note: Other options are available for the LOCALMOD command. See *z/VM: VMSES/E Introduction and Reference* for additional information.

7. If you want to use the New-Password-Phrase Exit (ICHPWX11), you must create a local modification to the RPIBLLPA build list. Otherwise, skip this step and continue with step 8.

```
LOCALMOD compname RPIBLLPA EXEC (MODID Lnnnn
```

   a. Reply **1** to the prompt message.
   b. In the displayed RPIBLLPA EXEC file, remove the asterisks (*) to uncomment the following section.

```
*:OBJNAME. ICHPWX11 LEPARMS RENT REUS LET NCAL XREF SIZE 100K,80K
*:OPTIONS. IGNORE
*:PARTID.  ICHPWX11 TXT
*:EOBJNAME.
```

   c. Enter **file** on the XEDIT command line to file your changes.

```
====>file
```

8. Build your new local modification on the test build disk.

```
SERVICE compname BUILD
```

9. Put the local modification into production.

```
PUT2PROD compname
```

   Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Modify Full-Part ASSEMBLE Files, TEXT Files, and Build List

During customization of RACF you might have to modify ASSEMBLE files and corresponding TEXT files that are serviced as full-part replacement.

The following instructions describe how to create and build the new part. Use the substitution command values provided in the instructions that pointed you to this topic.

1. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

   For *compname*, use:

   RACF
   For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

2. If you are only deleting a part from a build list, continue with step 5.
3. Create and apply a local modification to the ASSEMBLE file.

```
LOCALMOD compname fn ASSEMBLE (MODID Lnnnn
```

   a. Reply **1** to the prompt message.
   b. In the displayed file, make your local modification.
   c. Enter **file** on the XEDIT command line to file your changes.

```
====>file
```

Note: Other options are available for the LOCALMOD command. See *z/VM: VMSES/E Introduction and Reference* for additional information.

4. If you are not adding an object to or deleting an object from a build list, continue with step 6.
5. To add objects to or delete objects from a build list, create a local modification to the build list.

```
LOCALMOD compname blist EXEC (MODID Lnnnn
```

   a. Reply **1** to the prompt message.
   b. To add a new command part to the build list, add the following statements to the displayed build list at the end of the file.

      The following is an example of what you would add into a TXTLIB build list.

```
:OBJNAME. memname
:OPTIONS. NOGETLVL
:PARTID. memname TEXT
:EOBJNAME.
*
```

      Where *memname* is the file name of your new command TEXT file.

      The following is an example of what you would add into a LOADLIB build list.

```
:OBJNAME. memname LEPARMS RENT REUS LET NCAL XREF SIZE 100K,80K
:OPTIONS. NOGETLVL
:PARTID.  memname TEXT
:EOBJNAME.
*
```

      Where *memname* is the file name of your new command TEXT file.

   c. To delete an object from the build list, comment out the following lines from the build list. To comment out the lines, add an asterisk (*) at the beginning of each line.

      The following example corresponds to a LOADLIB build list.

```
:OBJNAME. ICHRCX02 LEPARMS RENT REUS LET NCAL XREF DCBS SIZE 100K,80K
:OPTIONS. CONCAT SYSLIB RACFOBJ
:PARTID.  ICHRCX02 TXT
:OPTIONS. ENTRY ICHRCX02
:EOBJNAME.
```

   d. Enter **file** on the XEDIT command line to file your changes.

```
====>file
```

6. Build your new local modification on the test build disk.

```
SERVICE compname BUILD
```

7. Put the local modification into production.

```
PUT2PROD compname
```

Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Modify Full-Part Replacement TEXT Files

During customization of RACF you might have to modify TEXT files that are serviced as full-part replacement using your existing ASSEMBLE file.

The following instructions describe how to create and build the new TEXT file. Use the substitution command values provided in the instructions that pointed you to this section.

1. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

For *compname*, use:

RACF
For installing on minidisks

RACFSFS
For installing in shared file system (SFS) directories

RACFPANL
For installing on minidisks with RACF ISPF panels

RACFPANLSFS
For installing in SFS directories with RACF ISPF panels

2. Create the ASSEMBLE file on the 2C2 disk.
3. Make your local modification to the copy of the ASSEMBLE file on your local modifications disk (LOCALSAM 2C2).
4. Assemble the file.

```
VMFASM fn SERVP2P compname ($SELECT OUTMODE LOCALSAM
```

Notes:

a. Other options are available for the VMFASM command. See *z/VM: VMSES/E Introduction and Reference* for additional information.

b. If the assemble function is successful, the file *fn* TXT00000 is placed on the LOCALSAM 2C2 (E) disk.

5. Rename the *fn* TXT00000 file on the 2C2 disk and log the local modification.

```
VMFREPL fn TEXT SERVP2P compname ($SEL LOGMOD Lnnnn OUTM LOCALSAM
```

6. Build your new local modification on the test build disk.

```
SERVICE compname BUILD
```

7. Put the local modification into production.

```
PUT2PROD compname
```

Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Modify Full-Part Replacement TEXT Files and Build List

During customization of RACF you might have to modify TEXT files that are serviced as full-part replacement using the existing ASSEMBLE file.

The following instructions describe how to create and build the new TEXT file. Use the substitution command values provided in the instructions that pointed you to this section.

1. Access the RACF service disks.

   ```
   VMFSETUP SERVP2P compname
   ```

   For *compname*, use:

   RACF
   For installing on minidisks

   RACFSFS
   For installing in shared file system (SFS) directories

   RACFPANL
   For installing on minidisks with RACF ISPF panels

   RACFPANLSFS
   For installing in SFS directories with RACF ISPF panels
2. If you are deleting an object from a build list, continue with step 9.
3. If you want to use the Password Encryption Exit (ICHDEX01) and you want to use the default IBM-supplied TEXT file already assembled, complete step 4, skip steps 5 through 7, and continue with step 8.
4. Copy the ICHDEX01 TEXT file to the local modifications disk (LOCALSAM 2C2).

   ```
   VMFREPL ICHDEX01 TEXT SERVP2P compname ($SEL LOGMOD Lnnnn OUTM LOCALSAM
   ```
5. Create or modify the ASSEMBLE file on the 2C2 disk.
6. Assemble the file.

   ```
   VMFASM fn SERVP2P compname ($SELECT OUTMODE LOCALSAM
   ```

   Notes:

   a. Other options are available for the VMFASM command. See *z/VM: VMSES/E Introduction and Reference* for additional information.

   b. If the assemble function is successful, the file *fn* TXT00000 is placed on the LOCALSAM 2C2 (E) disk.
7. Rename the *fn* TXT00000 file on the 2C2 disk and log the local modification.

   ```
   VMFREPL fn TEXT SERVP2P compname ($SEL LOGMOD Lnnnn OUTM LOCALSAM
   ```
8. If you are not adding an object to or deleting an object from a build list, continue with step 10.
9. To add an object to or delete an object from a build list, create a local modification to the build list.

   ```
   LOCALMOD compname blist EXEC (MODID Lnnnn
   ```

   a. Reply **1** to the prompt message.

   b. To uncomment an object in a build list, delete the asterisk (*) at the beginning of each line.

   The following example corresponds to a LOADLIB build list.

   ```
   *:OBJNAME. ICHDEX01 LEPARMS RENT REUS LET NCAL XREF DCBS SIZE 100K,80K
   *:OPTIONS. CONCAT SYSLIB RACFOBJ
   *:OPTIONS. INCLUDE RACFOBJ(ICHDEX01)
   *:OPTIONS. ENTRY ICHDEX01
   *:EOBJNAME.
   ```

c. To comment out an object in a build list, add an asterisk (*) at the beginning of each line.

   The following example corresponds to a LOADLIB build list.

```
:OBJNAME. ICHDEX01 LEPARMS RENT REUS LET NCAL XREF DCBS SIZE 100K,80K
:OPTIONS. CONCAT SYSLIB RACFOBJ
:OPTIONS. INCLUDE RACFOBJ(ICHDEX01)
:OPTIONS. ENTRY ICHDEX01
:EOBJNAME.
```

d. Enter **file** on the XEDIT command line to file your changes.

```
====>file
```

10. Build your new local modification on the test build disk.

```
SERVICE compname BUILD
```

11. Put the local modification into production.

```
PUT2PROD compname
```

Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Modify Full-Part Replacement Parts

During customization of RACF you might have to modify files that are serviced as full-part replacements (for example, EXEC files).

The following instructions describe how to create and build the new part. Use the substitution command values provided in the instructions that pointed you to this topic.

1. Access the RACF service disks.

```
VMFSETUP SERVP2P compname
```

   For *compname*, use:

   RACF
   For installing on minidisks

   RACFSFS
   For installing in shared file system (SFS) directories

   RACFPANL
   For installing on minidisks with RACF ISPF panels

   RACFPANLSFS
   For installing in SFS directories with RACF ISPF panels

2. Create and apply a local modification to the part.

```
LOCALMOD compname fn ft (MODID Lnnnn
```

3. Build your new local modification on the test build disk.

```
SERVICE compname BUILD
```

4. Put the local modification into production.

```
PUT2PROD compname
```

Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Build or Link-Edit a Library

During customization of RACF you might have created your own exits that need to be built or link-edited into a RACF library. You need to assemble the exit source file and build the text into a RACF library (for example, LOADLIB).

The following instructions describe how to build the appropriate library. Use the substitution command values provided in the instructions that pointed you to this topic.

1. Log on to the MAINT740 user ID.
2. Access the RACF service disks.

   ```
   VMFSETUP SERVP2P compname
   ```

   For *compname*, use:

   RACF
   For installing on minidisks

   RACFSFS
   For installing in shared file system (SFS) directories

   RACFPANL
   For installing on minidisks with RACF ISPF panels

   RACFPANLSFS
   For installing in SFS directories with RACF ISPF panels
3. Place the exit source ASSEMBLE file on the LOCALSAM 2C2 (E) disk and assemble the file.

   ```
   VMFASM fn SERVP2P compname (OUTMODE LOCALSAM
   ```
4. Build your new library on the test build disk.

   ```
   VMFBLD PPF SERVP2P compname blist memname (ALL
   ```
5. Put the new library into production.

   ```
   PUT2PROD compname
   ```

   Note: For information on using RACMAINT to test code before putting it into production, see "RACF Security Server" in Appendix C in *z/VM: Service Guide*.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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
US

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

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

IBM may use or distribute any of the information you provide 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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

© Copyright IBM Corp. 1993, 2025

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.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain 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 actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain 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.

Programming Interface Information

This publication primarily documents intended programming interfaces that allow the customer to write programs to obtain services of an external security manager.

This document also contains information that is NOT intended to be used as programming interfaces. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

NOT Programming interface information
End of NOT programming interface information

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

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.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

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:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

© Copyright IBM Corp. 1993, 2025

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_3.1.0/pdf/ickug00_v3r1.pdf), GC35-0033

Related Products

XL C®++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Index

Special Characters

*
  in the database name table 19

Numerics

382 abend code 120
383 abend code 111
385 completion code 122, 123

A

abend code
  382 120
  383 111, 112
access authority
  checking with RACHECK 118
access requests, CP disposition for 37
ACEE data area
  default built by RACINIT 110
  when ICHRIX01 is responsible for creating 112
ACIGROUP control statements 65
adding installation written commands 62
algorithms
  password encryption 29
ALign keyword
  IRRUT400 utility 95
allocation
  BAM/allocation comparison 81
AUDIT operand
  ADDSD command 7
  ALTDSD command 7
  RALTER command 7
  RDEFINE command
    effect on system performance 7
  SETROPTS command
    effect on system performance 7
authorization checking
  using global access checking 17

B

backout routines
  commands that have 144
backup RACF databases
  conditions for creating 5
  creating 5
  defining in the database name table 6, 19
  effect on RACF processing 2
  levels of backup 6
  maintaining 2
  performance impact 6
  point-in-time copy 85
  using the RVARY command 139
BAM blocks
  BAM/allocation comparisons 81
  encoded map
    codes used in 82
  sample printout of an encoded map 81
block alignment
  when using the IRRUT400 utility 90
buffers
  how the RACF manager keeps track of 21

C

cached auxiliary storage subsystem
  setup for a backup database 5
callers of exit routines
  summary of 108
CDT (class descriptor table)
  defining new classes 25
  generating the table 27
  list of the general resource classes for z/OS 155
  list of the general resource classes for z/VM 155
  names of IBM-supplied classes for z/VM systems 155
  when the RACF database is shared 25
change count in the ICB
  during processing on a shared RACF database 21
changing command names and syntax 59
class
  defining new classes 25, 26
  installation-defined
    deleting 27
class descriptors
  adding, modifying, and deleting 25
class names
  list of IBM-supplied general resource classes 155
command names and syntax
  changing 59
command output
  tailoring 59
command session, RACF, requiring password for 38
commands
  adding installation written 62
completion code
  385 122, 123
control statements
  for IRRUT100 utility 75
  for IRRUT200 utility 79
control unit
  for the RACF database 5
CP
  LOGONBY 148
  recovery procedures 148
CP, RACF modules residing in 41
creating backup databases
  using the IRRUT200 utility 5
  using the IRRUT400 utility 5
cross-reference report
  produced by IRRUT100 utility 72

CSTCONS (message routing table)
adding additional user IDs 54
example of 54
explanation of 54
updating 55

D

DASD data set
not allowing duplicate profile names on z/OS 158
data blocks
how resident blocks affect system performance 7
location of storage 21
size of 21
specifying in ICHSECOP 158
specifying resident blocks in the database name table 19, 21
data set name table
format of the flag field 19
database considerations
when the RACF database is shared on z/VM 3
database name table
modifying 22
database range table
customizing 22
description 23
modifying 24
database recovery
on z/OS 139
on z/VM 139
DES (data encryption standard) algorithm
replacing by using the ICHDEX01 exit routine 133
DES (data encryption standard) authentication option
using 29
description 156
device
for the RACF database 5
DF/DSS DEFRAG, how to use 4
DIRACC class
description 155
DIRECTRY class
description 155
DIRSRCH class
description 155
DSMON (data security monitor)
RACF exits report 107
DUPdatasets keyword
IRRUT400 utility 95
duplicate data set names
disallowing in ICHSECOP on z/OS 158
dynamic allocation parameters
in the ICHRSMFI module 35
dynamic parse
initializing 41

E

encoded map
of a BAM block 81
sample printout by IRRUT200 81
ENCRYPT keyword
ENCRYPT keyword *(continued)*
on ICHEACTN macro 133
on ICHETEST macro 133
encryption
exit routine 133
meaning of 31
EXECs
GENNUC EXEC 64
ICHDIRMV EXEC 63
ICHSFS EXEC 64
ICHSFSDF EXEC 61, 63
ISPF EXEC 63
provided on product tape 63
RAC EXEC 57, 63
RACALLOC EXEC 64
RACDSF EXEC 64
RACDSMON EXEC 63
RACFADU EXEC 63
RACFCONV EXEC 65
RACFDBU EXEC 64
RACFDEL EXEC 64
RACFLIST EXEC 63
RACFPERM EXEC 63
RACFSVRS EXEC 65
RACGROUP EXEC 63
RACINITD EXEC 64
RACIPLXI EXEC 65
RACONFIG EXEC 10, 21, 64
RACOUTP EXEC 64
RACRPORT EXEC 63
RACSETUP EXEC 64
RACSTART EXEC 65
RACSVRXI EXEC 65
RACUT100 EXEC 65
RACUT200 EXEC 65
RACUT400 EXEC 65
RCMDRFMT EXEC 64
RPIBLDDS EXEC 64
RPIDELU EXEC 64
RPIDIRCT EXEC 64
SMFPROF EXEC 63
exit routine
examining during RACF failures 139
for commands on z/OS 128
FRACHECK 124, 125
how they affect system performance 11
ICHCCX00 on z/OS 131
ICHCNX00 on z/OS 129
naming convention table on z/OS 127
new password 113
new password phrase 115
password encryption 133
possible uses of
modifying data set naming conventions on z/OS 128
RACDEF 121
RACF exits report from DSMON 107
RACHECK 119
RACINIT 111
RACLIST selection 127
report writer 135, 136
requirements for 107
SAF router 136
summary of callers 108

exit routine *(continued)*
use case
password quality control 115
exit routines
RPIPACEX 61
exits report
from DSMON 107
extending a database
using IRRUT400 utility 90

F

FACILITY class
description 155
failsoft processing
exits called 141
for RACDEF 158
how it can affect system performance 10
not active with SETRACF INACTIVE 141
z/OS users logged on 142
z/OS users not logged on 142
z/VM users not logged on 142
failures
during I/O operations on the RACF database 144
during RACF command processing 144
during RACF manager processing 147
failsoft processing 142
on the RACF service machine 148
recovery procedures 139
setup for a backup RACF database 5
use user ID in UADS to logon 141
FIELD class
description 155
field-level access checking
to control access to fields within a profile 13
FILE class
description 155
flags
flag field in the database name table 19
formatted output of the index blocks 80
FRACHECK macro
exit routines 124, 125
how FRACHECK processing affects system performance 17
use of resident profiles 17
z/VM does not use 17
FREESPACE keyword
IRRUT400 utility 95
FSOBJ class
description 155
FSSEC class
description 155

G

general resource
definition 24
general resource class
changing the class descriptor table 27
defining new classes 25
defining, modifying, or deleting with RACDEF 121
IBM-supplied 155
general resource class *(continued)*
list of those in the class descriptor table 155
generic profile
during authorization checking 17
internal name for the range table 23
modified by RACF on z/OS 23
performance considerations 17
GENNUC EXEC
brief description 64
GLBLDSK macro
using 52
global access checking
for bypassing normal RACHECK processing 17
global access table
using a global access table 17
GLOBAL class
description 155
global minidisk table, defining 38
GLOBALAUDIT operand
RALTER command
effect on system performance 7
GMBR class
description 155
group
information about provided by IRRUT100 utility 72
group name
listing all occurrences on the RACF database 72
GTERMINL class
description 155

I

I/O operations
failures during 144
ICB (index control block)
change counts 21
RBAs of the templates defined 81
ICH508I message 128
ICHCCX00 exit routine
callers of on z/OS 131
parameter list on z/OS 131
return codes on z/OS 132
uses of on z/OS 128
when entered on z/OS 129
ICHCNX00 exit routine
parameter fields available to on z/OS 129
parameter fields that can be changed on z/OS 129
parameter list 129
return codes on z/OS 130
uses of on z/OS 128
when entered on z/OS 128
ICHDEX01 exit routine
callers of 133
modifying on z/VM 134
parameter list 133
restrictions 133
return codes 133
uses for 133
ICHDIRMV EXEC
brief description 63
ICHERCDE macro
generating the class descriptor table 27

ICHNCV00 127
ICHNGMAX macro 39
ICHNRT00 routine
when called on z/OS
128
ICHPWX01 exit routine
conditions for gaining control 113
parameter list 113
requirements for 113
return codes 114
ICHPWX11 exit routine
parameter list 115
return codes 117
ICHRCX01 exit routine
parameter list 119
return codes 119
ICHRCX02 exit routine
parameter list 119
return codes 120
what RACF does before it receives control 112, 120
ICHRDSNT module
description 19
example of using 21
for specifying the RACF databases 6
modifying
database name table 22
selecting the number of resident data blocks 7, 21
ICHRDX01 exit routine
called during failsoft processing 141
return codes 122
ICHRDX02 exit routine
ICHRDX01 exit routine 121
ICHRDX02 exit routine 121
return codes 123
ICHRFR01 module
description 28
ICHRFR0X module
description 28
ICHRFRTB macro
entries corresponding to the class descriptor table
entries 25, 26
ICHRFX01 exit routine
environment executed in 124
parameter list 124
requirements for 123, 124
return codes 124
ICHRFX02 exit routine
environment executed in 125
parameter list 125
reason codes 125
requirements for 125
return codes 125
ICHRIN03 module
conversions before the exit routines receive control 107
ICHRIX01 exit routine
creating and initializing the ACEE 112
making password checks 115
parameter list 111
requirements for 111
return codes 111
ICHRIX02 exit routine
parameter list 111
requirements for 112
return codes 112
ICHRIX02 exit routine *(continued)*
what RACF does before it receives control 112
ICHRLX01 exit routine
requirements for 126
return codes 126
ICHRLX02 exit routine
parameter list 127
requirements for 126
return codes 127
ICHRRCDE CSECT name 25
ICHRRCDE module
deleting a class 27
list of class descriptors within 25
ICHRRNG module
correspondence to the database name table 23
description 22, 23
example 23
example of using 23
location 22
modifying
database range table 24
relationship to IRRUT400 output databases 95
ICHRSMFE exit routine
modifying on z/VM 136
parameter list 136
return codes 136
uses of 135
when it is called 136
ICHRSMFI module
changing 35
format of 35
ICHRSW1 exit routine
parameter list 135
ICHRTX00 exit 136
ICHRTX01 exit 136
ICHSEC00 module
processing of in-storage buffers 21
ICHSECOP module
bypassing RACF initialization processing 157
caution on z/VM 157
description of options 157
disallowing duplicate names for DASD data set profiles
on z/OS 158
format 157
resident data blocks
selecting the number 158
ICHSFS EXEC
brief description 64
ICHSFSDF EXEC
brief description 63
with the RAC EXEC 61
ICHUT400 utility
creating 5
with LOCKINPUT keyword 5
inactive (RACF)
by bypassing RACF initialization processing 157
index blocks
formatted output by IRRUT200 80
sample formatted output by IRRUT200 80
scanning with IRRUT200 utility 79
structure correction with IRRUT400 utility 90
unformatted output by IRRUT200 80
index compression
when using the IRRUT400 utility 90

index entries
problems due to failures during RACF manager processing 147
index structure
correcting when using IRRUT400 utility 90
initialization processing
bypassing 157
initialization routine
locating the naming convention table module on z/OS 128
Initializing a RACF database 45
installing RACF
formatting the RACF database 70
storage requirements 149
installing RACF on z/OS
formatting the RACF database 70
internal names
how RACF constructs for the range table 23
IPL
use of the ICHSECOP module 157
IRR402I message 147
IRR403I message 147
IRR404I message 147
IRRMIN00 utility
description 69
execs you need to execute 71
executing 71
group name or user ID 75
input 71
output 72
return codes 72
using 71
using on z/OS 71
IRRUT100 utility
associated exit routine 73
description 72
information provided by the report 72
sample output of the printed report 76
work CMS file on z/VM 73
work data set on z/OS 73
z/VM example 75
IRRUT200 utility
BAM/allocation comparison 81
description 76
examples of coding on z/OS 83
for copying the primary database to the backup 2
functions 76
identifying problems with the RACF database 147
input and output 78
prompts 78
sample output
formatted index blocks 80
of the encoded map 81
unformatted index blocks 80
scanning the index blocks 79
using 78
using a copy to update the RACF database 10
utility control statements 79
IRRUT300 utility
to correct problems with the RACF database 147
IRRUT400 utility
allowable keywords on z/OS and z/VM 94
description 89
examples of coding on z/OS 95
IRRUT400 utility *(continued)*
examples of coding on z/VM 97
executing
input 91
processing of conflicts and inconsistencies 95
return codes from the utility 104
extending a database 90
how it works 90
using with keyword NOLockinput 94
when not to use 95
with LOCKINPUT keyword 5
z/VM example of copying 97
z/VM example of splitting 97
ISPF data files
storage requirements 150
ISPF EXEC
brief description 63

J

JCL (job control language)
parameters ignored when bypassing RACF initialization on z/OS 157
JOB statement
parameters ignored when bypassing RACF initialization on z/OS 157

K

KDFAES (key derivation function with advanced encryption standard) authentication option
using 29

L

location of the RACF database 4
location of the RACF database on z/OS 4
LOCKinput keyword
IRRUT400 utility 94
LOCKINPUT keyword
using with IRRUT400 90
logging
how it affects system performance 7
using RACHECK exit routine to modify 118

M

mandatory access control (MAC)
filtering 9
resetting the MAC filter 9
master RACF database 19
message support 54
messages
ICH508I 128
IRR402I 147
IRR403I 147
IRR404I 147
produced by IRRUT200 utility 89
messages, suppressing 38
minidisk, moving 47
minidisks
moving 50
moving a minidisk 47

multiple RACF databases
using the IRRUT400 utility 89
multiple RACF service machines 39
multiple service machines
coordination of commands 57
dedicating for RACROUTE request processing 57
description 56
setting up 151
using 56
multiple services machines
CP directory statements 152
initializing 154
installing 151
multisystem environment
using the RVARY command 139

N

naming convention table
functions it should perform on z/OS 128
use of on z/OS 127
when processing occurs on z/OS 128
naming conventions
changing the standard naming conventions on z/OS 127
modifying with exits on z/OS 128
NGROUPS_MAX constant 39
NOALign keyword
IRRUT400 utility 95
NOCMDVIOL operand
SETROPTS command
effect on system performance 8
NODUPdatasets keyword
IRRUT400 utility 95
NOFREEspace keyword
IRRUT400 utility 95
NOLockinput keyword
IRRUT400 utility 94
non-shared RACF database
consideration when changing to shared on z/VM 3
processing of in-storage buffers 21
non-VSAM data set
formatting for use as a RACF database 70
NOSAUDIT operand
SETROPTS command
effect on system performance 8
NOTABle keyword
IRRUT400 utility 94

O

OpenExtensions for VM, activating RACF support for 39
operating system and RACF interaction 1
operator prompts
during failsoft processing 141
for an asterisk in the database name table 19
for the RVARY ACTIVE/INACTIVE command 141
for the RVARY SWITCH command 141
options
changing the ICHRSMFI module 35
customizing the RACF database
range table 22
options *(continued)*
defining resource classes 25
ICHDEX01 exit 29
password authentication 29
specifying RACF database options
database name table 19
ICHSECOP module 157
options available to RACF 19
OSIX constant NGROUPS_MAX 39
OWNER field
resolving conflicts with RACLIST selection exit routine 126

P

panels
RACALLOC 68
RACDSF 68
RACSEC 69
parameter lists
ICHCCX00 exit routine on z/OS 131
ICHCNX00 exit routine on z/OS 129
ICHDEX01 exit routine 133
ICHPWX01 exit routine 113
ICHPWX11 exit routine 115
ICHRCX01 exit routine 119
ICHRCX02 exit routine 119
ICHRFX01 exit routine 124
ICHRFX02 exit routine 125
ICHRIX01 exit routine 111
ICHRIX02 exit routine 111
ICHRLX02 exit routine 127
ICHRSMFE exit routine 136
ICHRSW1 exit routine 135
modifying with the SAF router exit routine 136
PassTicket
validating 31
password
checking validity with RACINIT 110
encrypting 31
encryption exit routine 133
new password exit routine 113
PassTicket as an alternative for 31
processing 32
quality control 115
PASSWORD command
invoking the new password exit routine 113
making password checks 115
performance
factors affecting the system 5
how exit routines affect 11
how FRACHECK processing affects 17
how logging affects 7
how RACF commands can affect 10
how RACHECK processing affects 17
how RACINIT processing affects 17
how statistics gathering affects 15
how utility programs affect 10
using resident index and data blocks 7
point-in-time backup copy 85
posit number 16, 26, 28
postprocessing exit routines
FRACHECK macro
environment executed in 125

postprocessing exit routines *(continued)*
FRACHECK macro *(continued)*
parameter list 125
reason codes 125
requirements for 125
return codes 125
RACDEF macro
return codes 123
uses for 121
RACHECK macro
parameter list 119
return codes 120
uses for 118
RACINIT macro
parameter list 111
requirements for 112
return codes 112
uses for 110
RACLIST macro
requirements for 126
return codes 126
preprocessing exit routines
command - ICHCCX00 on z/OS
return codes 132
FRACHECK macro
environment executed in 124
parameter list 124
requirements for 123, 124
return codes 124
how they affect system performance 11
ICHCCX00
parameter list 131
ICHCNX00
calling before IRRUT100 utility 73
parameter list on z/OS 129
return codes on z/OS 130
RACDEF macro
return codes 122
RACHECK macro
parameter list 119
return codes 119
RACINIT macro
parameter list 111
requirements for 111
return codes 111
RACLIST macro
requirements for 126
return codes 126
what RACF does before the exits receive control 107
primary RACF database
defining in the database name table 19
PROCESS class
description 155
profile
commands that do not modify 144
defining, modifying, or deleting with RACDEF 121
not allowing duplicate DASD data set names on z/OS 158
specifying in ICHSECOP on z/OS 158
PSFMPL class
description 155
PTKTDATA class
description 155
PTKTVAL class
description 155
public minidisks 52
public minidisks, defining 38

R

RAC command processor
description 57
how it works 58
in RACF service machines 61
modifying 59
using with SFS files and directories 61
using with the RACF service machine 61
RAC EXEC
brief description 63
RACALLOC EXEC
brief description 64
RACDEF macro
exit routines 121
preprocessing routine called during failsoft 141
when RACF initialization is bypassed 158
RACDSF EXEC
brief description 64
RACDSMON EXEC
brief description 63
RACF
bypassing initialization processing 157
customization 19, 159
exit routines
encryption 133
FRACHECK 123–125
ICHCCX00 on z/OS 128, 131
ICHCNX00 on z/OS 128, 129
naming convention table on z/OS 127
new password 113
new password phrase 115
RACDEF 121
RACF exits report from DSMON 107
RACF report writer 135
RACHECK 118, 119
RACINIT 110, 111
RACLIST 126
RACLIST selection 127
report writer 136
summary of callers 108
exploiting new function 61
ICHSECOP warning 157
interaction with the operating system 1
meeting security requirements 1
performance considerations 5, 17
recovery procedures 142
storage requirements
ISPF 150
RACF database 149
system libraries 149
virtual 150
utilities
IRRMIN00 69
IRRUT100 72
IRRUT200 76
IRRUT400 89
summary 67

RACF command session, requiring password for 38
RACF commands
  exit routines for
    ICHCCX00 on z/OS 131
    ICHCNX00 on z/OS 128, 129
  exit routines for on z/OS 128
  failures during RACF command processing 144
  how they can affect system performance 10
  that do not modify RACF profiles 144
  that have recovery routines 144
  that perform multiple operations 146
  that perform single operations 145
  writing your own 61
RACF database
  additional backup measures 2
  alternate databases 2
  backup databases 2
  commands that modify only one profile 145
  commands that perform multiple operations 146
  considerations when shared on z/VM 3
  customizing
    range table 22
  database considerations 1
  database name standardization on z/OS 3
  database name standardization on z/VM 3
  deactivating with the RVARY command 139
  discrepancies between profiles 144, 146
  encrypting the passwords and OIDCARD data 29
  failures during I/O operations 144
  failures during RACF manager processing 147
  FBA devices 4
  identifying inconsistencies with IRRUT200 76
  if shared when failing 140
  levels of backup 6
  location of the RACF database 4
  location of the RACF database on z/OS 4
  maintaining the 139
  master RACF database 19
  modifying
    database name table 22
    database range table 24
  multiple databases 2
  point-in-time copy 85
  RACF database volumes on z/OS 4
  records initialized for a new database 70
  recovery procedures 139
  recovery with UADS password 141
  restoring 142
  selecting control unit and device 5
  shared database considerations on z/VM 3
  specifying options
    database name table 19
    ICHSECOP module 157
  storage requirement
    approximation 149
  the effect of switching databases on performance 2
  using DF/DSS DEFRAG 4
  using resident index and data blocks 7
  using the RVARY command 139
  when to issue commands that update 10
  z/VM running native 3
RACF database,
  extending with IRRUT400 utility 90
RACF database, *(continued)*
  locating occurrences of a user ID or group name 72
  summary statistics by IRRUT200 82
  using IRRMIN00 to format 70
  using utilities on
    IRRMIN00 69
    IRRUT100 72
    IRRUT200 76
    IRRUT400 89
    summary 67
RACF database, initializing 45
RACF database, nonrestructured
  multiple databases 2
RACF exits report
  from DSMON 107
RACF LOADLIBS
  RACFLPA 22
  RPICDE LOADLIB 29
RACF LOADLIBS on z/VM
  RACFLINK 19
RACF LOGON BY
  description 51
RACF manager
  failures during processing 147
  processing of in-storage buffers 21
  return code of 28 23
RACF modules
  requirements for where they reside on z/VM will reside in RACFLINK 149
RACF modules residing in CP 41
RACF options 19
RACF report writer
  exit routine 135, 136
  format of the ICHRSMFI module 35
  list of functions 35
RACF service machine
  activating RACMAINT service machine 148
  applying service 148
  failures on z/VM 148
  resuming RACF service 148
RACF service machines, defining multiple 39
RACF service machines, defining user IDs for 38
RACFADU EXEC
  brief description 63
RACFCONV EXEC
  brief description 65
  used to convert RACF database templates 71
  used to execute IRRMIN00 71
RACFDBU EXEC
  brief description 64
RACFDEL EXEC
  brief description 64
RACFEVNT class
  description 155
RACFLIST EXEC
  brief description 63
RACFPERM EXEC
  brief description 63
RACFSVRS EXEC
  brief description 65
RACFVARS class
  description 155
RACGROUP EXEC
  brief description 63

RACHECK macro
exit routines
uses for 118
how RACHECK processing affects system performance 17
preprocessing routine called during failsoft 141
RACINIT macro
building the default ACEE 110
effect of RACINIT processing on system performance 17
exit routines
uses for 110
when RACF initialization is bypassed 157
RACINITD EXEC
used to execute IRRMIN00 71
RACIPLXI EXEC
brief description 65
RACLIST macro
exit routines 127
RACONFIG EXEC
brief description 64
using with the database name table 21
with IRRUT200 10
RACOUTP EXEC
brief description 64
RACROUTE
using RACROUTE and installation-defined classes 29
RACROUTE macro
invoking the SAF router 136
RACROUTE request processing
dedicating a RACF service machine 57
RACRPORT EXEC
brief description 63
RACSERV macro 38, 39
RACSETUP EXEC
brief description 64
RACSTART EXEC
brief description 65
RACUT100 EXEC
used to execute IRRUT100 75
RACUT400 exec 91
RACVERFY FILE 78
range table 22
RBA (relative byte address)
of a BAM block 81
of the templates defined in the ICB 81
RCMDRFMT EXEC
brief description 64
RCMDS load module 58
reason codes
from ICHRFX02 exit routine 125
recovery procedures
for failures during I/O operations on the RACF database 144
for failures during RACF command processing 144
for failures during RACF manager processing 147
for the RACF database 139
synchronization considerations 142
using UADS user ID 141
recovery routines
commands that have 144
REFRESH GENERIC operands
SETROPTS command 15
REFRESH RACLIST operands
SETROPTS command 14
relationship of RACF and the operating system
description of 1
renaming a user 51
replaceable modules
ICHRDSNT 19
ICHRRNG 22
ICHRSMFI 35
ICHSECOP on z/OS 157
report
produced by IRRUT100 utility 72
sample output from IRRUT100 76
report writer
default values in the ICHRSMFI module 35
exit routine 135
format of the ICHRSMFI module 35
list of functions 35
resident data blocks
how they affect system performance 7
specifying in ICHSECOP 158
specifying in the database name table 19, 21
resident index blocks
how they affect system performance 7
resident profiles
use by FRACHECK for authorization checking 17
ways used by RACF 120
resource class
defining classes 25
defining new classes 25
on z/OS 25
on z/VM 25
resource manager 1
return codes
28 from the RACF manager 23
from ICHCCX00 exit routine on z/OS 132
from ICHCNX00 exit routine on z/OS 130
from ICHDEX01 exit routine 133
from ICHPWX01 exit routine 114
from ICHPWX11 exit routine 117
from ICHRCX01 exit routine 119
from ICHRCX02 exit routine 120
from ICHRDX01 exit routine 122
from ICHRDX02 exit routine 123
from ICHRFX01 exit routine 124
from ICHRFX02 exit routine 125
from ICHRIX01 exit routine 111
from ICHRIX02 exit routine 112
from ICHRLX01 exit routine 126
from ICHRLX02 exit routine 127
from ICHRSMFE exit routine 136
from IRRMIN00 utility 72
from IRRUT400 utility 104
router table
adding an entry 28
Router table
ICHRFR01 module 28
ICHRFR0X module 28
search order 28
RPICDE LOADLIB 29
RPIDELU EXEC
brief description 64
RPIDIRCT EXEC
brief description 64

RPIRAC module 58
RPIRACEX exit
  description 61
  how to use 61
RPIRCMTB
  ASSEMBLE file 62
  TEXT file 62
RPITMPTB
  ASSEMBLE file 62
  TEXT file 62
RVARSMBR class
  description 156
RVARY command
  ACTIVE operand 141
  deactivating the RACF database 139
  INACTIVE operand 141
  using in a multisystem environment 139
  using RVARY ACTIVE/INACTIVE 139
  using RVARY SWITCH 139
RVARYPW operand
  SETROPTS command
    defining passwords for RVARY 140

S

SAF (system authorization facility)
  invoking the SAF router 136
SAF router
  exit routine 136
SAF router exit routine
  how it receives control 136
  on MVS/XA 136
  uses for 136
scanning index blocks
  formatted printout of by IRRUT200 80
  unformatted printout of by IRRUT200 80
  with IRRUT200 utility 79
SCDMBR class
  description 156
SECDATA class
  description 156
SECLABEL auditing
  related system overhead 9
SECLABEL class
  description 156
security requirements
  how RACF meets 1
service machines
  coordination of commands 57
  dedicating for RACROUTE request processing 57
  description 56
  setting up 151
  using 56
service machines, RACF 38, 39
services machines
  CP directory statements 152
  initializing 154
  installing 151
SETRACF INACTIVE
  effect on failsoft processing 141
SETROPTS command
  operands
    GENLIST 12
    RACLIST 12
SETROPTS command *(continued)*
  REFRESH GENERIC operands 15
  REFRESH RACLIST operand 14
  specifying rules for passwords 115
SFS
  file pool server
    restrictions 52
  producing SMF records 8
  service machines
    dedicating 11
  SFSAUTOACCESS option 52
SFSAUTOACCESS 52
SFSCMD class
  description 156
shared file system (SFS)
  file pool server
    restrictions 52
  producing SMF records 8
  service machines
    dedicating 11
  SFSAUTOACCESS option 52
shared RACF database
  caution for class descriptor tables 25
  consideration when changing to non-shared on z/VM 3
  processing of in-storage buffers 21
  specifying the resident data block option 21
SMF data
  when restoring a RACF database 142
SMF records
  when ICHRSMFE exit routine is called 136
SMFPROF EXEC/idxterm>
  brief description 63
SORT/MERGE parameters
  in the ICHRSMFI module 35
specifying options
  switchable 2
statistics
  by IRRUT200 about the index 80
  by IRRUT200 about the RACF database 82
  how they affect system performance 15
  on the RACF backup database 6
  updating on the backup RACF database 20
storage requirements
  ISPF 150
  RACF database 149
  system libraries 149
  virtual for RACF 150
SURROGAT class
  description 156
SWITCH operand on the RVARY command 140
switchable RACF databases
  effect on RACF processing 2
switching RACF databases
  using the RVARY command 139
switching to the backup service machine 148
synchronization
  when restoring a database 142
  when restoring a RACF database 142
  when using RVARY in a multisystem environment 139
SYSSEC macro 37, 38
system libraries
  storage requirements 149
system performance
  factors affecting 5

system performance *(continued)*
how exit routines affect 11
how failsoft processing can affect 10
how FRACHECK processing affects 17
how logging affects 7
how RACF commands can affect 10
how RACHECK processing affects 17
how RACINIT processing affects 17
how statistics gathering affects 15
how utility programs affect 10
using resident index and data blocks 7

T

TABLE keyword
IRRUT400 utility 94
tailoring command output 59
TAPEVOL class
description 156
templates
verifications by IRRUT200 utility 81
TERMINAL class
description 156
tracks
number required for ISPF 150
number required for the system libraries 149

U

UAUDIT operand
ALTUSER command
effect on system performance 7
undefined users
supplying a user ID 110
UNLockinput keyword
IRRUT400 utility 94
user IDs
listing all occurrences on the RACF database 72
moving 50
users
information on provided by IRRUT100 utility 73
using LOGON BY
system programming 51
utilities
for use on the RACF database 67
how they can affect system performance 10
IRRMIN00 69
IRRUT100 72
IRRUT200 76
IRRUT400 89
utility control statements
for IRRUT200 utility 79

V

virtual storage requirements for RACF 150
VMBATCH class
description 156
VMCMD class
description 156
VMDEV class 156
VMLAN class
description 156
VMMAC class
description 156
VMMDISK class
description 156
VMNODE class
description 156
VMPOSIX class
description 156
VMRDR class
description 156
VMSEGMT class
description 156
VMSES/E 25
VMXEVENT class
description 156
VXMBR class
description 156

W

work data set for IRRUT100
format of the records 73
WRITER class
description 156

Z

z/VM event
logging 9
VMXEVENT class 9
z/VM systems
RACF database 1

Product Number: 5741-A09

Printed in USA

SC24-6312-74