SC28-1343-1
File No. S370-34

System Programming Library: Resource Access Control Facility (RACF)

Program Product

IBM

SC28-1343-1
File No. S370-34

Program Product

System Programming Library: Resource Access Control Facility (RACF)

Program Number 5740-XXH

Second Edition (December, 1985)

This is a major revision of SC28-1343-0. See the Summary of Amendments following the Contents for a summary of the changes that have been made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 1 Release 7, of the program product RACF (Resource Access Control Facility - Program Number 5740-XXH), and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. The previous edition still applies to RACF Version 1 Release 6 and may now be ordered using the temporary number ST28-1343. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments can be addressed to IBM corporation, Information Development, Department D58, Building 921-2, PO Box 390, Poughkeepsie, N.Y., 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication contains information for Version 1 Release 7 of the Resource Access Control Facility (RACF) Program Product (5740-XXH). It is intended for the use of system programmers or installation personnel responsible for:

- Installing RACF in the system
- Maintaining the RACF data sets
- Writing, testing, and installing the RACF exits
- Modifying the RACF program product to satisfy the installation's particular needs

The readers of this publication must be familiar with the information in the *RACF General Information Manual* and *RACF Security Administrator's Guide*. The *RACF Auditor's Guide*, which describes the RACF report writer, might also be useful.

The chapters in this publication are:

Chapter 1: Installing RACF - includes information on installing and modifying RACF.

Chapter 2: Operating Considerations - contains information needed for the on-going activities of a system with RACF installed.

Chapter 3: Performance Considerations - describes the factors that affect system performance.

Chapter 4: Recovery Procedures - describes backup measures for the RACF data set and procedures for recovering from failures that occur during RACF manager processing or the execution of RACF commands and other programs that use the RACF data set.

Chapter 5: RACF Options - describes RACF options and how to use them.

Chapter 6: RACF Utilities - describes the RACF utilities and how to use them.

Chapter 7: RACF Installation Exits - includes descriptions of the RACF installation exits and how to write exit routines.

Chapter 8: RACF Data Set - describes the format of a RACF data set.

Chapter 9: Storage Estimates - identifies the storage requirements for the RACF data set, the system libraries, the ISPF data sets, and the RACF program product.

Chapter 10: RACF Macros - describes the ICHERCDE, ICHRFRTB, ICHNCONV, ICHEINTY, ICHETEST, and ICHEACTN macros.

Chapter 11: SMF Records - includes the formats of the RACF SMF records (type 80 and 81) and the reformatted SMF records.

Chapter 12: RACF Data Areas - maps the ACEE, CGRP, DSDT, ISP, MDEL, RCVT, and RRPF data areas.

RACF Publications

These RACF publications are suggested for use with this book. If a shortened title is used in this publication, the short title appears in parentheses following the publications number.

- *Resource Access Control Facility (RACF): General Information Manual*, GC28-0722, which contains an overview of RACF and planning information for Version 1 Release 7.

- *Resource Access Control Facility (RACF) Security Administrator's Guide*, SC28-1340, which explains RACF concepts and describes how to plan for and implement RACF.

- *Resource Access Control Facility (RACF) Command Language Reference*, SC28-0733, which contains the functions and syntax of all the RACF commands.

  The *RACF General User Command Reference Card*, SX28-0609, contains information extracted from SC28-0733.

- *Resource Access Control Facility (RACF) Messages and Codes*, SC38-1014, which contains the RACF messages, the routing and descriptor codes, the RACF manager return codes, and the RACF-related system completion codes.

- *Resource Access Control Facility (RACF) Auditor's Guide*, SC28-1342, which describes auditing considerations, as well as how to use the RACF report writer and the data security monitor.

- *Resource Access Control Facility (RACF) User's Guide*, SC28-1341, which explains how to perform common end user tasks.

- *Resource Access Control Facility (RACF) Program Logic Manual*, LY28-0730, which describes the internal logic and organization of RACF. Three microfiche cards complement the program logic manual:

  Resource Access Control Facility (RACF) Data Areas, LYB8-0770

  Resource Access Control Facility (RACF) Macro Usage Table, LYB8-0888

Resource Access Control Facility (RACF) Symbol Usage Table, LYB8-0889

The RACF macros that are shipped with MVS are documented in the following publications:

- *OS/VS2 System Programming Library: Supervisor*, GC28-1046
- *OS/VS2 MVS Supervisor Services and Macro Instructions*, GC28-1114
- *MVS/Extended Architecture System Programming Library: System Macros and Facilities Volume 1*, GC28-1150
- *MVS/Extended Architecture System Programming Library: System Macros and Facilities Volume 2*, GC28-1151
- *MVS/Extended Architecture Supervisor Services and Macro Instructions*, GC28-1154

The publication *Resource Access Control Facility (RACF) Installation Reference Manual*, SC28-0734, applies to RACF Version 1 Release 5, and is no longer updated. Applicable information previously in SC28-0734 now appears in SC28-1340, SC28-1342, and SC28-1343.

The following publications contain detailed information about the RACF/VM Support PRPQ (Program Number 5767-002), which allows you to use RACF with VM systems:

- *Introduction to the Resource Access Control Facility/VM Support PRPQ*, GC34-2297. which contains overview and planning information on the use of RACF with a VM system.
- *Resource Access Control Facility/VM Support PRPQ Reference Manual*, SC34-2296, which identifies and explains the differences between the functions of RACF with MVS and VM systems.
- *RACF/VM Support PRPQ General User Command Reference Card*, SX22-0008, which contains the functions and syntax of the RACF commands on a VM system, as well as some examples of using the commands.

The following RACF self-study courses are available from SRA (Science Research Associates Incorporated):

- *RACF for the Security Administrator*, 32187
- *RACF for Technical Support Personnel*, 32188
- *RACF for the Auditor*, 32189

The *Catalog of IBM Education*, G320-1244, contains more information on these self-study courses.

In addition to the RACF publications, see the documentation for the products that you are using with RACF for additional RACF-related information.

After the general availability of RACF Version 1 Release 7, you can use the following temporary order numbers to order publications for RACF Version 1 Release 6.

- *Resource Access Control Facility (RACF) General Information Manual*, GT00-1731
- *Resource Access Control Facility (RACF) Security Administrator's Guide*, ST28-1340
- *System Programming Library: Resource Access Control Facility (RACF)*, ST28-1343
- *Resource Access Control Facility (RACF) Auditor's Guide*, ST28-1342
- *Resource Access Control Facility (RACF) Command Language Reference*, ST00-1732
- *Resource Access Control Facility (RACF) Program Logic Manual*, LT00-1733

Other Publications

These publications are referenced in the text and/or contain information related to using RACF. If a shortened title is used in this publication, the short title appears in parentheses following the publication number.

MVS/370 JCL User's Guide, GC28-1349

MVS/370 JCL Reference, GC28-1350

MVS/Extended Architecture JCL User's Guide, GC28-1351

MVS/Extended Architecture JCL Reference, GC28-1352

OS/VS2 System Programming Library: System Management Facilities (SMF), GC28-1030 (*System Management Facilities (SMF)*)

MVS/Extended Architecture System Programming Library: System Management Facilities, GC28-1153 (*System Management Facilities (SMF)*)

OS/VS2 System Programming Library: Initialization and Tuning Guide, GC28-1029 (*Initialization and Tuning*)

MVS/Extended Architecture System Programming Library: Initialization and Tuning, GC28-1149 (*Initialization and Tuning*)

OS/VS2 System Programming Library: Job Management, GC28-0627

MVS/Extended Architecture System Programming Library: System Modifications, GC28-1152

Operator's Library: OS/VS2 MVS System Commands, GC28-1031 (*System Commands*)

MVS/Extended Architecture System Commands, GC28-1206 (*System Commands*)

OS/VS2 System Programming Library: Data Management, GC26-3830

MVS/Extended Architecture System Programming Library: Data Management, GC26-4010

OS/VS2 MVS Data Management Services Guide, GC26-3875

MVS/Extended Architecture Data Management Services, GC26-4013

OS/VS Tape Labels, GC26-3795

MVS/Extended Architecture Tape Labels, GC26-4003

OS/VS Checkpoint/Restart, GC26-3784

MVS/Extended Architecture Checkpoint/Restart, GC26-4012

OS/VS Utilities, GC26-3902 (*Utilities*)

MVS/Extended Architecture Utilities, GC26-4018 (*Utilities*)

OS/VS Sort/Merge Programmer's Guide, SC33-4035

OS/VS2 System Programming Library: TSO, GC28-0629 (*System Programming Library: TSO*)

MVS/Extended Architecture System Programming Library: TSO, GC28-1173 (*System Programming Library: TSO*)

OS/VS2 TSO Command Language Reference, GC28-0646 (*TSO Command Language Reference*)

MVS/Extended Architecture TSO Command Language Reference, GD23-0259 (*TSO Command Language Reference*)

Data Facility Data Set Services General Information, GC26-3947 (*DFDSS General Information*)

Data Facility Data Set Services User's Guide and Reference, SC26-3949 (*DFDSS User's Guide and Reference*)

Device Support Facilities User's Guide and Reference, GC35-0033 (*DSF User's Guide and Reference*)

Contents

Figures

New Functions in RACF Version 1 Release 7

RACF Function	Supported by MVS/XA	MVS/370	VM	Other Requirements
Access control to load modules	Y	N	N	MVS/XA 2.1.2 or later, with a PTF
Program access to data sets	Y	N	N	MVS/XA 2.1.2 or later, with a PTF; and DFP 2.1.0 with a PTF
DASD erase-on-scratch	Y	N	N	DFP 2.1.0 with a PTF
Expanded RACF/CICS security support	Y	Y	N	CICS/OS/VS Version 1 Release 7
User or terminal time/day-of-week control	Y	Y	Y	
Realtime violation notification	Y	Y	Y	
Tape data set protection	Y	N	N	DFP 2.1.0 with a PTF
Tape bypass label (BLP) processing control	Y	N	N	DFP 2.1.0 with a PTF
Security classification of users and data	Y	Y	Y	
Data security monitor report enhancements	Y	Y	N	
Data security monitor authorization enhancements	Y	N	N	MVS/XA 2.1.2 or later, with a PTF
RVARY command authorization enhancement	Y	Y	Y	
Data set protect-all option	Y	Y	N	always-call support
Control of REVOKE/RESUME by date	Y	Y	Y	
Class descriptor table and RACF router table split	Y	Y	Y	
Logging OPERATIONS authority	Y	Y	Y	
Virtual storage constraint relief	Y	Y*	N	
Enhanced support for RACF ISPF panels	Y	Y	Y	ISPF Version 1 or 2 and TSO/E Release 2 or later
LISTDSD and RLIST enhancements	Y	Y	Y	
ADDSD and RDEFINE modeling enhancements	Y	Y	Y	
RACINIT without generating statistics	Y	Y	Y	

* Applies to systems with IMS/VS or CICS/VS (more efficient storage of RACF user data)

Term	Meaning
MVS/XA	MVS/System Product Version 2
MVS/370	MVS/System Product Version 1 Release 3 or later
VM	VM/System Product Release 3 or later, with or without the High Performance Option Release 3.2 or later, when used in conjunction with the RACF/VM Support PRPQ
DFP	Data Facility Product
ISPF	Interactive System Productivity Facility

Summary of Amendments

Summary of Amendments
for SC28-1343-1
for Resource Access Control Facility

This revision contains updates in support of RACF Version 1 Release 7. For a summary of the changes for RACF 1.7 see "New Functions in RACF Version 1 Release 7."

Editorial and technical changes have also been made throughout this major revision.

Summary of Amendments
for SC28-1343-0
as updated April 30, 1984
by Technical Newsletter SN28-0979

This newsletter contains updates in support of the RACF data security monitor. The changes are:

- A new topic, "Invoking the Data Security Monitor," in Chapter 2, contains sample JCL for executing the data security monitor program.
- A new topic, "RACF Exits Report," in Chapter 7 contains information about the exits report that the data security monitor produces.
- The space requirement for SYS1.LINKLIB is 54 tracks.

In addition, minor technical and editorial changes have been made.

Chapter 1. Installing RACF

To install RACF, you must understand the RACF system requirements as well as the following tasks:

- Preparing your system by allocating additional storage, if required
- Modifying the RACF program product package by:
  - Adding installation-written exit routines or a naming convention table.
  - Activating the Data Encryption Standard (DES) algorithm for encrypting passwords and OIDCARD data
  - Replacing the module (ICHSECOP) used to bypass RACF initialization processing, specify the number of resident index blocks, and disallow RACF protection for duplicate data set profile names
  - Replacing the started procedures module (ICHRIN03) that associates the names of started procedures with RACF userids and group names.
  - Creating multiple RACF data sets, which includes modifying the range table and creating a data set name table
  - Specifying the number of resident index/data blocks in the data set name table (ICHRDSNT)
  - Changing the RACF authorized-caller table (module ICHAUTAB)

This chapter contains general installation procedures for both new and previous RACF users. The RACF program directory that comes with the program product package contains detailed instructions on how to install and remove the RACF functions. It also contains information about sample JCL procedures provided with the RACF package. These procedures will help you to install RACF.

System Requirements

You can use RACF Version 1 Release 7 with OS/VS2 MVS/System Product Version 1 Release 3 and later as well as with MVS/System Product Version 2.

In addition, the following requirements depend on your environment:

- To use the RACF report writer function, install the DFSORT program (program number 5740-SM1) or an equivalent.
- To use the ISPF panels for RACF, install the Interactive System Productivity Facility Version 1 (ISPF, program number 5668-960) or ISPF Version 2 (ISPF, program number 5665-319) and TSO Extension (TSO/E) Release 2 (program number 5665-285). The use of the Interactive System Productivity Facility/Program Development Facility (ISPF/PDF, program number 5665-268) is recommended, but not required. (Without ISPF/PDF, when you want to edit a CLIST generated by the SEARCH command, the TSO (line) editor is invoked. The ISPF/PDF (fullscreen) editor is available only with ISPF/PDF.
- RACF is supported by TSO in a current TCAM and VTAM environment.
- To use RACF for IMS/VS user identification and resource authorization, install IMS/VS Version 1 Release 1.5 (program number 5740-XX2) or later.
- To use RACF for CICS/VS user identification and transaction authorization, install CICS/VS Version 1 Release 5 (program number 5740-XX1) or later.
- To use tape data set protection for non-VSAM data sets, install Version 2 Release 1 of Data Facility Product (DFP) and a PTF.
- To use the erase-on-scratch facility for non-VSAM data sets, install Version 2 Release 1 of Data Facility Product (DFP).
- To use the erase-on-scratch facility for VSAM data sets, install Version 2 Release 1 of Data Facility Product (DFP) and a PTF.
- To use the program control facility, install Version 2 Release 1.2 of MVS/System Product and a PTF.

Preparing Your System

Before installing RACF, review the related storage requirements. See Chapter 9, "Storage Estimates," for information about the storage requirements for the RACF data set, system library storage requirements for the RACF modules, and virtual storage required by RACF. See the RACF program directory for information about the DLIB system library requirements.

Consider the following storage requirements:

- Be sure you have sufficient space for the RACF data set(s). Allocation for the data set(s) will be done later during the installation procedure.

- Allocate additional space in system libraries, if required.

- Allocate additional space in virtual storage, if required. See *Initialization and Tuning* for the allocation procedures.

- Allocate additional space on the SMF data set and change your SMF reporting routines. See *System Management Facilities (SMF)*.

Migration and Coexistence

RACF Version 1 Release 7 is designed so that an installation can easily migrate to it from Release 6 by executing the RACF Data Set Initialization Utility Program (ICHMIN00) to update the templates in the RACF data set. Following the conversion, you can use the two releases of RACF concurrently or alternately, because either release of RACF will work properly with the converted RACF data set.

However, because of the enhanced capabilities of RACF Release 7, you must be aware of a few migration considerations. Consider the following items when using Release 6 and Release 7 concurrently on a shared system, or alternately during a migration period:

- The tape data set protection option is not available in RACF Version 1 Release 6, but, when a tape data set is protected with Release 7, the tape volume is also protected as in Release 6. However, the data set profile commands in Release 6 should not be used on the profiles created by the tape data set protection option, nor should a RACDEF-DELETE be issued to delete a TAPEVOL (or DATASET) profile without also deleting the corresponding DATASET (or TAPEVOL) profile. Each installation must consider the possible conflicts that might arise in its own environment.

- TSO IKJPARSE routines allow a user to abbreviate a keyword on a TSO command to the least number of characters that cause that keyword to be distinguished from all other keywords. With the addition of many keywords to the RACF command in Release 7, it is possible that some conflicts in abbreviations may arise. We strongly recommended that commands that are hard coded (for example, in programs and CLISTs) specify all keywords fully spelled out. In an interactive environment, abbreviations are acceptable because the parse routines have the ability to prompt the user when any conflicts arise.

- RACF Release 1 Version 6 ignores some options, specifically the following:

  - RACINIT time/day of week control
  - REVOKE/RESUME by date

Installing RACF for the First Time

The RACF program directory contains instructions on how to install RACF. In general, the procedure for installing RACF Release 7 when you have not had a previous RACF release installed consists of the following:

1. Install the RACF modules.

   Install RACF on a copy of your operating system for testing. You cannot test the RACF system until the entire installation procedure is complete and RACF is activated by an IPL.

2. Allocate, catalog, and format a RACF data set.

   RACF provides programs to format RACF data sets. See "RACF Data Set Initialization Program (ICHMIN00)" and "The RACF Data Set Split/Merge/Extend Utility Program (ICHUT400)" in Chapter 6 for a description of the programs, the requirements for running the programs, and the contents of the RACF data sets after being formatted.

3. Create a data set name table or alter the MSTRJCL (optional).

   You can define the RACF data set by creating a RACF data set name table (ICHRDSNT). For multiple RACF data sets, modify the IBM-provided range table (ICHRRNG) and use it in conjunction with the data set name table.

   As an alternative, you can define the RACF data set using MSTRJCL. However, by using the data set name table and the range table, you avoid having to update the MSTRJCL and you centralize control over RACF. See "Creating Multiple RACF Data Sets" later in this chapter.

4. Update TSO APF-authorized command and program tables.

   Update the TSO APF-authorized command table in SYS1.LPALIB with the APF-authorized RACF commands and update the APF-authorized program table with the names of the RACF programs. See the information on authorized program execution in *System Programming Library: TSO.*

5. Update the RACF class descriptor table and the RACF router table (optional).

   With the exception of DATASET, GROUP, and USER classes, all resource classes are represented in a class descriptor table (CDT) in two load modules

ICHRRCDE and ICHRRCDX. ICHRRCDE is optional and contains installation-defined resource classes while ICHRRCDX contains IBM entries. See "General Resource Classes" and "Class Descriptor Table," in Chapter 5 for information on the class descriptor table.

You should code an ICHRFRTB macro instruction for each entry added to the class descriptor table that will be accessed by the RACROUTE macro instruction. The ICHRFRTB macro generates entries in the optional installation-defined RACF router table, ICHRFR01. This table controls the action taken by the RACF router when it is invoked by the RACROUTE macro instruction. See "Router Table Entry Definition" in Chapter 5 for information on the RACF router table.

6. Create a started procedures table (optional).

   If you identify resources to be protected by RACF that are also accessed by started procedures, you must determine if changes are required for the started procedures replaceable module (ICHRIN03) that is part of the RACF program product. See Chapter 5 "RACF Options" for information on when entries are needed in the started procedures module and how to code the module.

   The started procedures table is a discrete load module. Within it, you can indicate that selected started procedures are to be considered as privileged, meaning that all RACHECKs are accepted unconditionally. In addition, the table may include a generic entry.

7. Create exit routines and/or a naming convention table.

   You can write exit routines for user transparency and for modifying your installation's RACF security measures. See "RACF Installation Exits."

   You can set up and enforce data set naming conventions that are different from the standard RACF naming conventions by using a naming convention table (ICHNCV00). For information on how to code the table, see "Naming Convention Table" in Chapter 7 and "ICHNCONV Macro" in Chapter 10.

8. Change the parmlib FIX list (optional).

   To ensure good IMS/CICS performance, several of the RACF load modules must reside in fixed LPA. A sample IEAFxx member is provided in member RACINSTL in SYS1.SAMPLIB. See "RACF Virtual Storage Requirements" in Chapter 9 for detailed information. Either create or alter the IEAFIXxx list in SYS1.PARMLIB (see *Initialization and Tuning*) or use the DATASET macro at SYSGEN time.

9. Install the RACF ISPF panels (optional).

   This procedure is optional depending on whether you have ISPF Version 1 (program number 5668-960) or later and intend to use the panels. To use the RACF ISPF panels, perform the following steps:

   - Optionally copy RACF ISPF panel driver load module ICHSPF00 (including any aliases) from LINKLIB to LPALIB for improved performance.

   - Concatenate the ISPF RACF panel libraries to the appropriate ISPF filenames in your TSO logon procedures. You must concatenate the following libraries:

     - RACF ISPF library SYS1.HRFPANL to the ISPF library associated with file name ISPPLIB

     - SYS1.HRFSKEL to ISPSLIB

     - SYS1.HRFMSG to ISPMLIB

     - SYS1.HRFCLST to SYSPROC.

   - Modify an existing ISPF panel (the utility selection menu, ISRUTIL, is recommended) to provide entry into the RACF selection menu. If your installation uses the TSO session manager and you want your users to have the option of either using the session manager or the new scrollable table support, use panel ICHPOOSM. If you do not want your users to use the session manager, use panel ICHP00.

   - If your installation does not have ISPF/PDF (Interactive System Productivity Facility/Program Development Facility, program number 5665-268) installed, you should modify panel ICHP192, as documented in the panel itself, to make use of the TSO editing facilities. You should also modify panel ICHH192 (the help panel) to say TSO edit instead of PDF edit.

10. IPL.

    The last step of the installation procedure is to IPL your system with the create link pack area (CLPA) parameter to initialize the system with RACF and to define the first basic profiles on each RACF data set.

    The IPL will do the following:

    - Define the RACF SVC entry points.

    - Load the pageable link pack area (PLPA) with the RACF code and TSO tables. This action is the the result of specifying the CLPA parameter.

    - Allow the master scheduler initialization exit routine to initialize the RACF control blocks (for example, the RCVT).

- Set the RACF options (for example, disallowing duplicate data set names).
- Write a record (type 81) to the SMF data set indicating the completion of RACF initialization.
- Define the following basic profiles on the new RACF data set:
  - IBMUSER (user profile)
  - SYS1, VSAMDSET, and SYSCTLG (group profiles, where VSAMDSET and and SYSCTLG are subgroups of SYS1)
  - Three connect profiles connecting IBMUSER to the three groups

During IPL, if the RACF data set cannot be found in the data set name table or in MSTRJCL, (that is, if the data set name table has not been created, or contains an * for the data set name, or the RACF data set is not cataloged), the system prompts the operator for a RACF data set name. The operator can respond with the correct name or with 'NONE'.

- If the operator specifies 'NONE' for all the RACF data sets, RACF is not active for this IPL. The effect is the same as bypassing RACF initialization processing by using the option in ICHSECOP.
- If the operator specifies 'NONE' during the ***first*** IPL after a RACF system has been installed, the basic profiles are not defined to RACF. They are defined during the first IPL with RACF active.

Other System Considerations

If you plan to make some of the changes listed below, make the changes before you define your users, groups, and resources to RACF:

- Add TSO profiles to the UADS data set for any new TSO users with the ADD subcommand of the TSO ACCOUNT command (see *System Programming Library: TSO*).
- Update the SYS1.HELP data set with the RACF commands member, using the IEBUPDTE system utility or the TSO EDIT command. See *Utilities* or *TSO Command Language Reference.*
- Assign a specific console to receive system security messages (message routing code = 9), using the VARY console command. See *System Commands.*

Installing RACF When You Have a Previous RACF Release

The RACF program directory contains instructions on how to install RACF. In general, the procedure for installing RACF Release 7 when you have a previous RACF release installed consists of the following:

1. Install the RACF modules.

   The recommended procedure is to install RACF Release 7 on a copy of your operating system for testing. You cannot test the RACF system until the entire installation procedure is complete and RACF is activated by the re-IPL.

2. Convert the RACF data set templates.

   You must convert the RACF data set templates to Release 7 by running ICHMIN00 (supplied with RACF 1.7) prior to an IPL with RACF active.

   See "RACF Data Set Initialization Utility Program (ICHMIN00)" in Chapter 6 for a description of the program, the requirements for running the program, and the contents of the RACF data set after the conversion is completed.

3. Create/update the data set name table or alter the MSTRJCL (optional).

   You can define the RACF data set by creating a RACF data set name table (ICHRDSNT). For multiple data sets, modify the IBM-provided range table (ICHRRNG) and use it in conjunction with the data set name table.

   As an alternative, you can define the RACF data set using MSTRJCL. However, by using the data set name table and the range table, you avoid having to update the MSTRJCL and you centralize control over RACF. See "Creating Multiple RACF Data Sets" later in this chapter.

   If you have a data set name table, you might want to update the table to use the resident data block option. This option allows both index and non-index blocks to be resident in storage, instead of just index blocks.

4. Update the RACF class descriptor table and the RACF router table (optional).

   With the exception of DATASET, GROUP, and USER classes, all resource classes are represented in a class descriptor table (CDT) in two load modules ICHRRCDE and ICHRRCDX. ICHRRCDE is optional and contains installation defined resource classes while ICHRRCDX contains IBM entries. If you have previously modified ICHRRCDE, you must build a new version containing you own entries. See "General Resource Classes" and "Class Descriptor Table," in Chapter 5 for information on the class descriptor table.

   You should code an ICHRFRTB macro instruction for each entry added to the class descriptor table that will be accessed by the RACROUTE macro instruction. The ICHRFRTB macro generates entries in the optional installation-defined RACF router table, ICHRFR01. This table controls the action taken by the RACF router when it is invoked by the RACROUTE macro instruction. If you have previously modified ICHRFR01, you must build a new version containing you own entries. See "Router Table Entry Definition" in Chapter 5 for information on the RACF router table.

5. Create/update the started procedures table (optional).

   If you identify resources to be protected by RACF that are also accessed by started procedures, you must determine if changes are required for the started procedures replaceable module (ICHRIN03) that is part of the RACF program product. See Chapter 5 "RACF Options" for information on when entries are needed in the started procedures module and how to code the module.

   The started procedures table is a discrete load module. Within it, you can indicate that selected started procedures are to be considered as privileged, meaning that all RACHECKs are accepted unconditionally. In addition, the table may include a generic entry.

6. Create installation exits and/or a naming convention table.

   You can write installation exits for user transparency and for modifying your installation's RACF security measures. The RACF installation exits are described in Chapter 7 "RACF Installation Exits."

   You can set up and enforce data set naming conventions that are different from the standard RACF naming conventions by using a naming convention table (ICHNCV00). For information on how to code the table, see "Naming Convention Table" in Chapter 7 and "ICHNCONV Macro" in Chapter 10.

7. Change the parmlib FIX list (optional).

   To ensure good IMS/CICS performance, several of the RACF load modules must reside in fixed LPA. See "RACF Virtual Storage Requirements" in Chapter 9 for detailed information. Either create or alter the IEAFIXxx list in SYS1.PARMLIB (see *Initialization and Tuning*) or use the DATASET macro at SYSGEN time.

   Note: Several modules involved with IMS authorization checking have moved from LPALIB to LINKLIB and require changes to any IEAFxx member.

8. Install the RACF ISPF panels (optional).

   This procedure is optional depending on whether you have ISPF Version 1 (program number 5668-960) or later and intend to use the panels. To use the RACF ISPF panels, perform the following steps:

   - Optionally copy RACF ISPF panel driver load module ICHSPF00 (including any aliases) from LINKLIB to LPALIB for improved performance.

- Concatenate the ISPF RACF panel libraries to the appropriate ISPF filenames in your TSO logon procedures. For this option, concatenate the following libraries:

  - RACF ISPF library SYS1.HRFPANL to the ISPF library associated with file name ISPPLIB
  - SYS1.HRFSKEL to ISPSLIB
  - SYS1.HRFMSG to ISPMLIB
  - SYS1.HRFCLST to SYSPROC.

- Modify an existing ISPF panel (the utility selection menu, ISRUTIL, is recommended) to provide entry into the RACF selection menu. If your installation uses the TSO session manager and you want your users to have the option of either using the session manager or the new scrollable table support use panel, ICHPOOSM. If you do not want your users to use the session manager, use panel , ICHP00.

- If your installation does not have ISPF/PDF (Interactive System Productivity Facility/Program Development Facility, program number 5665-268) installed, you should modify panel ICHP192, as documented in the panel itself, to make use of the TSO editing facilities. You should also modify panel ICHH192 (the help panel) to say TSO edit instead of PDF edit.

- If you previously had RACF 1.6 installed and if you deleted ICHDDX01 from LPALIB to implement DES encryption, you must delete the module again.

9. IPL.

   The last step of the installation procedure is to IPL your system. Specify the create link pack area (CLPA) parameters for this IPL.

Modifying the RACF Program Product Package

This section contains information on how to put RACF functional modifications on your system. All the modifications are optional.

Adding Installation-Written RACF Exit Routines

The RACF preprocessing and postprocessing exit routines and a naming convention table can be added to your system or changed at any time. However, the new routines do not become effective until after the next IPL with the CLPA option specified.

You can use SMP procedures or the linkage editor to put the load modules for the exit routines or the naming convention table into SYS1.LPALIB.

For details on how to write RACF exit routines, see Chapter 7, "RACF Installation Exits." For details on how to create a naming convention table, see "Naming Convention Table" in Chapter 7.

Activating the Data Encryption Standard Algorithm

To provide greater protection than the masking algorithm used in RACF releases prior to Release 6, RACF uses the Data Encryption Standard (DES) algorithm to encrypt the passwords and operator identification card (OIDCARD) data.

RACF has a dummy encryption installation exit, ICHDEX01, that unconditionally returns with a return code of 4, which causes RACF to use the masking algorithm for password and OIDCARD data. To activate the DES algorithm, you must delete this dummy exit from SYS1.LPALIB and perform an IPL with the CLPA option specified, or delete it from the distribution library before installing RACF.

If you assign a new password (or OIDCARD data) using the DES algorithm, it is not possible for the user to log on to a RACF release prior to Release 6 using that password. To permit the user to log on, you must reset the user's password. There is no problem in logging onto Release 7 with a password that was assigned under a prior RACF release. For more information, see "Using the Data Encryption Standard (DES) Algorithm" in Chapter 5.

Replacing the ICHSECOP Module

You can replace the ICHSECOP module anytime after RACF has been installed. This module contains option control data areas and flags that can be used to bypass RACF initialization processing during IPL, specify the number of resident index blocks, and disallow duplicate data set profile names. For details of how to code this module, see Chapter 5, "RACF Options."

You can use SMP procedures (the ZAP statement) or the linkage editor to set these options in module ICHSECOP. The load module is ICHSEC00, which resides in SYS1.LINKLIB. If you use the linkage editor to replace the module, you should use the load module configuration for ICHSEC00 that is provided in the RACF program product package.

The new options are not effective until after the next IPL.

Replacing the Started Procedures Module

Started procedures have system-generated JOB statements that do not contain the USER, GROUP, or PASSWORD parameter. However, only users and groups can be specifically authorized to access RACF-protected resources. To give started procedures the same ability, you can associate the names of started procedures with RACF userids and group names. This option is part of a process that allows started procedures, such as JES, to have specific authorization to access RACF-protected resources, such as spool data sets.

Before replacing the started procedures module, ensure that each group name and userid to be specified in the new module is defined to RACF. If a started procedure mentioned in the module accesses any resources that are already RACF-protected, you must be sure that the equivalent userid or group name is authorized to access each of the RACF-protected resources. (Use the LISTGRP, LISTUSER, LISTDSD, RLIST, ADDGROUP, ADDUSER, and PERMIT commands as needed.) See the *Command Language Reference* for descriptions of the commands. For details on how to write the started procedures module, see Chapter 5, "RACF Options."

The started procedures module (ICHRIN03) is a separate load module. The RACF initialization module (ICHSEC00) locates the load module and stores its address in the RACF CVT. If RACF cannot locate the module, the system issues message ICH512I, and RACF is not activated.

The new module is not effective until after the next IPL with the CLPA option specified. (You could, however, also load the started procedures module into the MLPA, so that the linkpack area does not have to be recreated.)

Creating Multiple RACF Data Sets

Whether you are creating a RACF data set for the first time or updating an existing data set, you must run the RACF data set initialization program ICHMIN00. See Step 1 in Figure 1-1.

If you feel that a single RACF data set will serve your purposes adequately, you can disregard Step 2. However, if you have decided to take advantage of multiple RACF data sets, you must now modify the IBM-provided range table in module ICHRRNG and create the data set name table, ICHRDSNT. (See "Range Table" and "Data Set Name Table" in Chapter 5 for the formats of these tables.)

The range table and the original RACF data set are used as input to the ICHUT400 split/merge/extend utility program. This program divides the input data set into the number of RACF data sets you specified in the range table (see Step 2 in Figure 1-1).

At IPL time (Step 3 in Figure 1-1), the master scheduler initialization routine (MSI) gathers information from the data set name table, the range table, and the new RACF data sets and constructs a data set descriptor table. The RACF manager uses this data set descriptor table, which resides in the common storage area (CSA), to maintain and process the data sets.

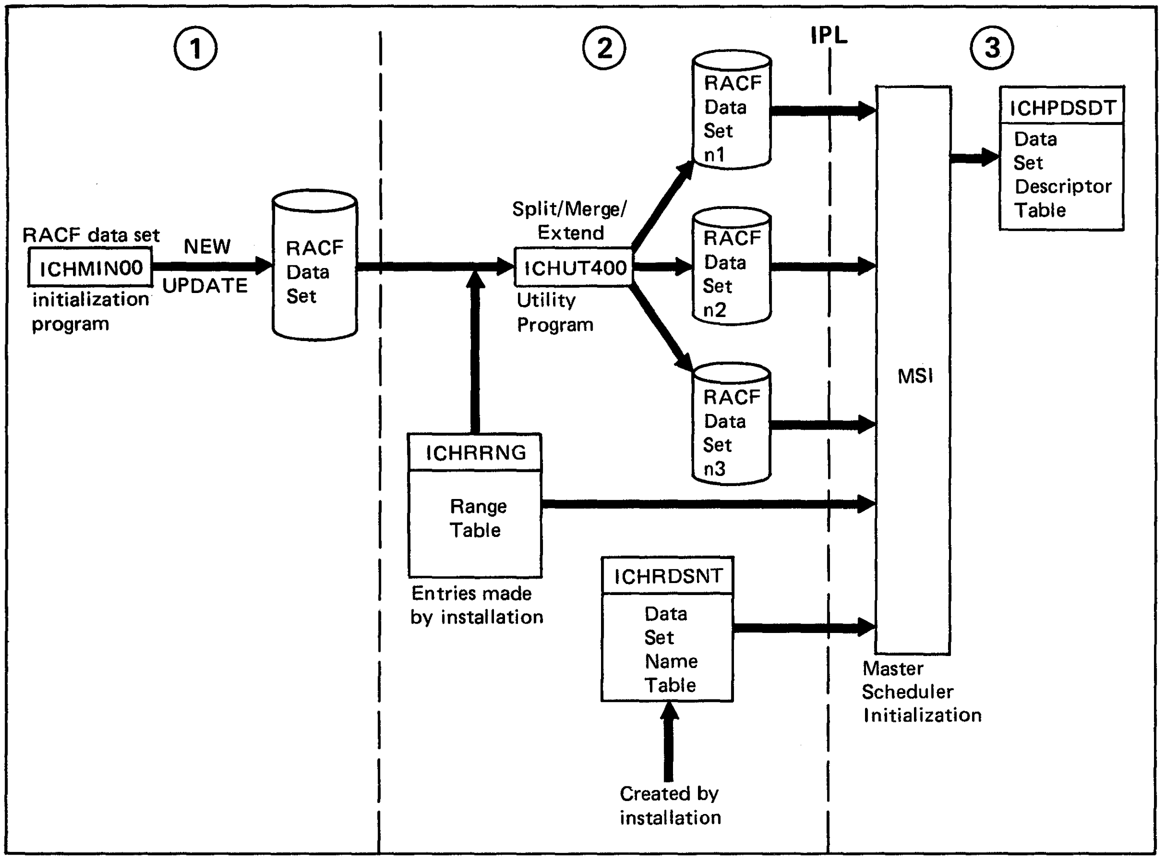


Figure 1-1. Creating Multiple RACF Data Sets

Specifying the Number of Resident Index/Data Blocks

The data set name table (ICHRDSNT) is an installation-written load module that describes the RACF data sets to RACF. This table may reside in SYS1.LINKLIB or any other library concatenated to SYS1.LINKLIB via the LNKLSTxx member in SYS1.PARMLIB.

In the data set name table, you can specify the number of resident index blocks to be made resident in the common service area (CSA). Or, you can specify the number of resident data blocks for each primary RACF data set. If you select the resident data block option, all types of data blocks (profile and block availability mask (BAM) blocks as well as index blocks) are kept in the CSA.

For more information on the data set name table, see “Data Set Name Table” in Chapter 5.

Changing the RACF Authorized-Caller Table

A program that invokes a RACF SVC routine (RACDEF, RACHECK, RACINIT, or RACLIST) must be authorized to issue the call. A program is authorized if it executes in supervisor state or in system key 0 - 7, or if it is authorized by the authorized program facility (APF). (For more information on APF, see *SPL: Supervisor* or *SPL: System Macros and Facilities.*) In addition, a program is authorized to invoke the RACINIT SVC (without the NEWPASS keyword) and/or the RACLIST SVC if that program is fetched from an authorized library and its program name exists in the RACF authorized-caller table.

Figure 1-2 shows this RACF SVC authorization scheme more clearly.

	To issue a RACF SVC				
The calling program must be one of the following:	RACDEF	RACHECK (See note)	RACINIT with NEWPASS keyword	RACINIT without NEWPASS keyword	RACLIST
APF-authorized	X	X	X	X	X
Executing in supervisor state	X	X	X	X	X
Executing in system key 0-7	X	X	X	X	X
Fetched from an authorized library and named in a RACF authorized-caller table				X	X
Note: RACHECK requires authorization only when the PROFILE, CSA, ACEE, or LOG parameters are specified.					

Figure 1-2. RACF SVC Authorization

The RACF authorized-caller table contains the names of programs that your installation authorizes to issue the RACLIST SVC or the RACINIT SVC without the NEWPASS keyword.

The RACF authorized-caller table resides in the link pack area (LPA) in ICHAUTAB. ICHAUTAB is an installation-replaceable module. To add an entry to the RACF authorized-caller table, you can do one of the following:

- Use the SPZAP service aid to add the entry to the IBM-supplied ICHAUTAB module.

  Note: ICHAUTAB can handle up to six table entries. If your installation requires more than six, you must reassemble the ICHAUTAB module.

- Re-assemble the ICHAUTAB module with the new entry and linkedit it again into the LPA. (Verify that the module is reentrant.)

For more information on changing the RACF authorized-caller table, see "Changing the ICHAUTAB Module" in Chapter 5.

Installation management must ensure that the programs it includes in the RACF authorized-caller table execute in a controlled environment; that is, users cannot modify code in these programs without prior review of the code by installation management. Also, installation management should use RACF to protect their authorized libraries to ensure that only authorized users linkedit programs into these libraries.

Chapter 2. Operating Considerations

This chapter describes considerations that installation personnel must be aware of in the operation of a system that has RACF installed.

The resources that RACF can protect are divided into two categories: DASD data sets and general resources. General resources includes DASD volumes and tape volumes. The following sections describe considerations related to using RACF with these resources. The sections are:

- DASD data sets
- DASD volumes
- Tape data sets
- Tape volumes
- RACF data set considerations
- Other considerations

DASD Data Sets

This section describes:

- Using utilities on RACF-protected DASD data sets - lists the system utilities for which RACF performs authorization checking and discusses some special rules that you must consider when using utilities on RACF-protected data sets.

- Moving a RACF-indicated DASD data set between systems - gives the possible situations and some considerations when moving RACF-indicated data sets from one system to another.

- Using access method services commands - describes using access method service commands with RACF-protected VSAM data sets.

Also see the section "DASD Volumes."

Using Utilities on RACF-Protected DASD Data Sets

RACF performs authorization checking for RACF-protected DASD data sets that are accessed by the following system utilities when the utilities issue the OPEN macro instruction:

ADRDSSU
ICKDSF
IEBCOMPR
IEBEDIT
IEBGENER
IEBISAM
IEBUPDTE
IEHATLAS
IEHLIST

IEBCOPY
IEBDG
IEBPTPCH
IEBTCRIN
IEHMOVE

To use these utilities on RACF-protected data sets, you must be defined to RACF and must be permitted access to any RACF-protected data sets that the utilities access (unless the UACC for the resource is sufficient to allow access).

Notes:

1. *You can use standard or nonstandard naming conventions when you define DASD data set profiles to RACF. By default (the standard naming convention), RACF expects the high-level qualifier of the name of a data set profile to be either a RACF-defined userid or group name.*

   RACF also allows you to use options to modify existing data set names to make them conform to RACF standard naming conventions. For example, the single-level name prefixing facility of RACF adds a qualifier to make the data set name acceptable to RACF routines. If you are not familiar with the options for nonstandard naming conventions, see the Security Administrator's Guide for more information.

   You also have the ability to create a naming conventions table (ICHNCV00), which RACF uses to check the data set name in all commands and SVCs that process data set names. Creating this table will help you set up and enforce data set naming conventions that are different from the standard RACF naming convention. For more information, see "Naming Convention Table" in Chapter 7 and "ICHNCONV Macro" in Chapter 10.

2. *See Chapter 6, "RACF Utilities," for a description of the RACF utilities that can be used on the RACF data set.*

The following topics describe special rules you must consider when using utilities on RACF-protected data sets.

Using Utilities with the OPERATIONS or group-OPERATIONS Attribute

A user who has the OPERATIONS attribute can:

- Create, rename, and define group data sets for groups to which the user is not connected, and does not have ALTER authority to the name if it is protected by a generic profile on an always call system. (If the user with the OPERATIONS attribute is connected to a group with less than CREATE authority, then the user may not perform any of these functions.)

- Create, rename, and define user data sets that are prefixed by another user's userid (unless, for rename, specifically prohibited in the data set's access list).

Thus, a user with the OPERATIONS attribute can perform many operations on DASD data sets using the system utilities.

The group-OPERATIONS user can perform all operations that can be performed by the OPERATIONS user; however, the authority is limited to the resources that are within the scope of the group to which the user is connected with the group-OPERATIONS attribute. For more information on the scope of the group, see the *Command Language Reference.*

Renaming RACF-protected Data Sets

You can rename a DASD data set that is protected by a discrete or generic RACF profile using the IEHPROGM utility, the access method services ALTER command, or the TSO RENAME command. IEHMOVE can also rename a data set but does so by creating a new data set having the new name. The following rules apply when renaming a data set:

- You cannot rename a multivolume, non-VSAM data set for which a discrete profile exists.
- You must have the OPERATIONS attribute (or group-OPERATIONS with the restrictions it carries) or have ALTER access authority to the data set.

  Note: If the data set is protected by a discrete profile, you cannot rename the data set to a name whose high-level qualifier is a group that you are connected to with less than CREATE authority, regardless of your OPERATIONS or group-OPERATIONS attribute.
- You must have the same authority to the new name as would be required to create it.
- The new name must conform to the RACF data set naming conventions, unless the naming convention table modifies the processing of data set names.
- If the data set is covered by a generic profile, you cannot rename it unless the new name is also covered by a generic profile and you have either ALTER authority to both new and old generic profiles or the OPERATIONS attribute (or group-OPERATIONS attribute with the restrictions it carries).
- You cannot rename an individual data set of a GDG if:
  - It is protected by a profile for the base portion of the GDG name.
  - The new name is a non-GDG name or is a GDG name for which there is no base profile defined.

To effectively rename a data set that cannot be renamed using IEHPROGM or TSO RENAME because of the above restrictions, copy the data set (using IEHMOVE) to one having the new name.

When you rename a data set that is protected by a discrete profile, RACF makes the following changes to the profile:

- If you do not have the OPERATIONS attribute (or group-OPERATIONS with the restrictions it carries) and the new name indicates a user data set (that is, the high-level qualifier is a userid), the access list for the data set remains the same, but the profile is changed to show you as the owner.

If you have the OPERATIONS attribute (or group-OPERATIONS with the restrictions it carries), the user whose userid is the high-level qualifier of the renamed data set becomes the owner.

In both cases, the profile changes to show the current connect group as the one under which the data set was renamed.

- If you have the GRPACC attribute and the high-level qualifier of the old data set name is a group name, RACF removes the group name from the access list.

  Note: If the high-level qualifier of the new data set name is also a group name, RACF adds that group name to the access list in Step 3. This action occurs even if the same group was removed in this step.

- If the new name indicates a group data set (the high-level qualifier is a group name), RACF updates the access list in the following way. Your userid is added to the list and given ALTER authority, unless your userid is already in the list. In this case, your authority remains unchanged. If you have the GRPACC attribute, the group indicated by the new name is added to the list and given UPDATE authority. The profile is also updated to show you as the owner of the data set (unless your authority to rename the data set is through your OPERATIONS or group-OPERATIONS attribute, in which case the owner is not changed) and to show the current connect group as the one under which the data set was renamed.

Caution: No change occurs in generic profiles applying to a data set being renamed. As a result of being renamed, a data set might be protected by a different generic profile than applied to the old name.

Using IEHMOVE With the ADSP Attribute

The following rules apply when you use the IEHMOVE system utility with RACF-indicated DASD data sets and you have the ADSP attribute:

- Moved and copied data sets that follow the RACF naming convention for data sets are automatically defined to RACF for protection. (IEHMOVE fails if the new data set name does not follow the RACF naming convention.)

- You cannot create a data set whose name has a high-level qualifier that is not your own userid (unless you have the OPERATIONS or group-OPERATIONS attribute or the data set being moved or copied is a group data set and you are connected to that group with at least CREATE access authority).

- You cannot move a data set with a target volume specified that is the same as the originating volume unless you code RENAME on the MOVE statement. If you do not code RENAME, IEHMOVE fails when it attempts to allocate an IEHMOVE-generated name that does not follow RACF naming conventions.

- If you select the option that prevents data sets with the same names from being defined to RACF with discrete profiles (by modifying the ICHSECOP module), then IEHMOVE cannot move or copy a data set that is

RACF-protected with a discrete profile unless the new data set has a different name than the old data set.

Using IEHMOVE With the COPYAUTH Parameter

On the MOVE and COPY statements of the IEHMOVE system utility, you can specify the COPYAUTH parameter. (Note: You cannot move a data set with a target volume specified that is the same as the originating volume unless you code RENAME on the MOVE statement.) The COPYAUTH parameter enables you to use the discrete profile of the old RACF-protected data set as a model to build a discrete profile for and RACF-indicate the new data set.

This modeling capability causes RACF to copy directly the following fields:

- access lists
- level
- UACC
- warning and logging options (auditing flags),
- installation data
- security categories and security levels
- erase option indicator
- user to be notified

The owner (the content of the owner field) is determined by the following rules:

- If the current user does not have the OPERATIONS or group-OPERATIONS attribute, then the user becomes the owner of the data set profile.

- If the current user has the OPERATIONS or group-OPERATIONS attribute, then either:

  - For a new user data set that has a different high-level qualifier other than the modeled data set name, the user whose userid is the high-level qualifier of the new data set name becomes the owner.

  - In all other cases, IEHMOVE copies the owner field directly from the model.

Note: A data set that is not RACF-indicated will not be protected after moving unless there is suitable generic profile on the destination system (and always call is in use).

Using the DFDSS and DSF Utilities

The DFDSS (Data Facility Data Set Services) and DSF (Device Support Facility) utilities perform access authorization checking for RACF-protected DASD data sets at the volume and data set levels. If the user is authorized to access a volume with ALTER authority to the volume or with the OPERATIONS user attribute, then access authorization is not performed for each data set on the volume. This feature is particularly useful for operations personnel, because they do not need individual authorization to access all of the data sets on a volume. It is also more efficient because only one RACINIT needs to be done.

During a RESTORE operation, DFDSS will define profiles for the RACF-indicated data sets being restored to RACF provided these data sets were defined to RACF with discrete profiles before the DUMP operation. The one exception to this rule occurs during a full volume RESTORE. When performing a full volume restore, DFDSS does not create profiles for any RACF-indicated data sets being restored. Therefore, if RACF profiles do not exist for these data sets, they will be inaccessible until RACF profiles are built for them.

The following access authorities are required in order to use DFDSS on RACF-protected data sets and volumes:

Compress PDS: UPDATE authority to volume in DASDVOL class or UPDATE to data set

Copy or dump with delete: ALTER authority to volume in DASDVOL class or ALTER to data set

Copy or dump without delete: READ authority to volume in DASDVOL class or READ to data set

RESTORE or COPY to preallocated data set: UPDATE authority to volume in DASDVOL class or UPDATE to target data set

RELEASE: ALTER authority to volume in DASDVOL class or ALTER to data set

PRINT: READ authority to volume in DASDVOL class or READ to data set

DEFRAG: UPDATE authority to volume in DASDVOL class or UPDATE to data set

The following access authorities are required in order to use DFDSS on RACF-protected full volumes:

Input volume: READ authority to volume in DASDVOL class

Output volume: ALTER authority to a volume in DASDVOL class data set RESTORE or for a partial COPY or RESTORE

The DSF utility protects RACF-protected data sets as follows:

- If the NOPURGE parameter is specified in the INIT command, the command terminates if RACF-protected data sets are found on the volume.

- If PURGE is specified in the INIT command, then DSF checks whether the user has either volume authority (by virtue of the OPERATIONS attribute or ALTER authority to the volume) or ALTER authority for all RACF-protected data sets on the volume.

For more information, see *DFDSS General Information*, *DFDSS User's Guide and Reference*, and *DSF User's Guide and Reference*.

Moving a RACF-Indicated DASD Data Set Between Systems

You can move a RACF-indicated DASD data set from system to system. Four situations are possible. You can move the data set from a system with RACF active to:

- Another RACF-active system (an MVS system with RACF installed and active, and with a RACF data set different from the one used by the source system)

  Note: If the source and destination systems share the same RACF data set, then no special action is needed to protect the data set.

- A RACF-inactive system (an MVS system with RACF installed but with the bypass RACINIT option in effect or with RACF deactivated by the RVARY command)

- A non-RACF system with RACF indicator checking (an OS/VS2 system with either Supervisor Performance #2 (SU7) or Data Management (SU8) installed but RACF not installed)

- A non-RACF system (a system with neither SU7 nor SU8 installed)

  Note: In this situation, the data set is not protected in any way on the non-RACF system.

Moving a RACF-indicated Data Set to a RACF-Active System

When a RACF-indicated data set is moved to a system with RACF active, the data set might or might not be defined to RACF on the destination system. If the data set is not defined to RACF, you must define it on the destination system and issue the ADDSD command with the NOSET operand. You specify NOSET because the data set is already RACF-indicated. If the data set is already defined to RACF on the destination system, no additional steps are needed; the data set is fully RACF-protected.

Caution: The access lists should be identical; otherwise, an integrity exposure may exist.

You can move a RACF-indicated data set to a system that already has a RACF-defined data set with the same name. If the data sets reside on volumes with different serial numbers, issue the ADDSD command with the NOSET operand to separately define the data set. If they reside on volumes with the same serial number, the data sets share the same discrete profile. There is only one access list and one set of statistics and logging options.

Regardless of whether the data set is RACF-indicated, if its name matches a generic profile at a destination system that has RACF active and generic profile checking enabled, the data set will automatically be protected if always call is in effect. The generic profile that matches at the destination system can have attributes (such as an access list) totally different from the discrete or generic profile that applied to the data set at the source system.

Moving a Data Set with a Discrete Profile to a RACF-Inactive System

When a RACF-indicated data set that is protected by a discrete profile is moved to a system with RACF inactive, attempts to access the data set on the destination system will cause RACF to ask the operator to allow access to a resource.

If you want access to a resource without the operator intervening, you can issue the DELDSD command on the source system (before moving the data set) to turn off the data set's RACF indicator and to delete the data set's RACF profile. RACF protection for the data set no longer exists.

Moving a RACF-indicated Data Set to a Non-RACF System With RACF Indicator Checking

When a RACF-indicated DASD data set is moved to a non-RACF system, attempts to access the data set on the destination system fail with a system E82 ABEND code. To prevent the ABEND, take one of the following actions:

- For either a VSAM or a non-VSAM data set, issue the DELDSD command on the source system (before moving the data set) to turn off the data set's RACF indicator and to delete the data set's profile.

- To turn off the RACF indicator for a non-VSAM data set but preserve the profile, perform the following steps on the source system before the move:

  1. Issue the ALTDSD command with the ADDVOL and NOSET operands to add a dummy volume number to the data set's RACF profile.

  2. Issue the ALTDSD command with the DELVOL and SET operands to delete the volume number of the data set from the RACF profile and turn off the RACF indicator. (The dummy volume number remains in the profile.)

  3. When you move the data set back to the source system, issue the ALTDSD command with the ADDVOL and SET operands to restore the volume number in the profile and to set the RACF indicator. Then issue ALTDSD with the DELVOL and NOSET operands to delete the dummy number.

Notes:

1. *There is no way to turn off the RACF indicator for a VSAM data set and still preserve its discrete profile.*

2. *For both a VSAM data set and a non-VSAM data set, the data set will have RACF protection only if generic access checking is enabled and a generic profile applies; otherwise, it reverts to password protection if the data set is password protected.*

Moving a Multivolume RACF-indicated Data Set Between Systems

When moving a multivolume RACF-indicated DASD data set from a source system to a designation system, you might need to update the RACF data set on the destination system. This step is necessary if you extend a non-VSAM data set to additional volumes on one system and then move it to another system where the data set is defined to RACF. In this case, the RACF data set on the destination system does not have the new volume serial number in the data set profile. The procedure required to add the new volume to the profile depends on whether the data set was extended on a RACF-active system or a non-RACF (or RACF-inactive) system.

Note: This explanation assumes the data set is RACF-defined on the destination system. If it is not, you must issue the ADDSD command to define it.

If the data set was extended on a RACF-active system, you must issue the ALTDSD command (with the ADDVOL and NOSET operands) on the destination system. This command adds the new volume serial number to the RACF data set profile but does not change the data set's RACF indicator.

Note: The source system automatically added the new volume number to its own RACF data set profile when the data set was extended. At the same time, the source system set the RACF indicator for that portion of the data set that is on the new volume.

If the data set was extended on a non-RACF or RACF-inactive system, you must issue the ALTDSD command (using the ADDVOL and SET operands) on the destination system. In addition to adding the new volume serial number to the RACF data set profile, this command sets the RACF indicator for that portion of the data set on the new volume. You use the SET operand because the non-RACF source system did not set the indicator when the data set was extended.

Multivolume VSAM data sets do not require these procedures. The RACF profile for a VSAM data set gives the volume serial number of only the catalog containing the data set entry. Therefore, a VSAM data set can extend to additional volumes without requiring changes to the RACF profile. Also, the VSAM catalog contains one RACF indicator for the entire data set regardless of the number of volumes; an indicator does not need to be set for each new volume.

Using Access Method Services Commands

This section describes considerations when using the following access method services commands with RACF-protected VSAM data sets:

- LISTCAT
- REPRO
- RESETCAT
- IMPORT
- IMPORTRA

LISTCAT Command

When you use the LISTCAT command on a RACF-protected VSAM data set and you have less than ALTER access authority to the data set, you might receive an authorization failure message followed by listed information. It is likely that you requested a list of passwords, which requires ALTER access authority. VSAM writes the error message that indicates you do not have sufficient authority to list passwords and then lists the requested information (except passwords).

REPRO/RESETCAT/IMPORT/IMPORTRA Commands

A discrete data set profile in the RACF data set contains the volume serial number of the catalog for a RACF-protected VSAM data set. This volume serial number must match the volume serial number supplied on the RACHECK macro instruction or RACF cannot locate the correct profile in the RACF data set. (VSAM processing routines supply the volume serial number of the containing catalog on the RACHECK macro instruction.)

The REPRO, RESETCAT, IMPORT, and IMPORTRA commands can cause the volume serial number of the catalog containing a RACF-protected data set to change and differ from the volume serial number in the data set's profile in the RACF data set. These commands do not invoke RACF to update the profile in the RACF data set.

Therefore, the user who is maintaining the RACF-protected data set must update the data set profile with the correct volume serial number.

To update data set profiles, issue the ALTDSD command with the ALTVOL operand. Note that you can use the SEARCH command to build a command procedure containing ALTDSD commands for all VSAM data sets cataloged on the same volume.

Or, to update profiles, you can use the TSO command procedure facility to: (1) obtain a list of the catalog entries for a volume for RACF-indicated data sets by using the LISTCAT command, (2) obtain a list of the RACF-indicated VSAM data sets by using the SEARCH command, and (3) compare the volume serial numbers of RACF-indicated data sets (obtained by the LISTCAT command) with the volume serial numbers of RACF-indicated data sets (obtained by the SEARCH command) to ensure the correctness and completeness of volume serial number information.

DASD Volumes

This section describes considerations when using the RACF DASDVOL authorization facility to authorize selected users and groups to DASD volumes that contain RACF-protected data sets.

Scratching DASD Data Sets

A user who has ALTER access authority to a DASD volume can scratch data sets on the volume whether or not the user is authorized access to the data sets. (If the user is not authorized to access the data set, RACF issues message ICH408I to the security console to report an access violation even though the data set is scratched.) When a data set is scratched, RACF deletes the discrete profile for the data set from the RACF data set or, in the case of a multivolume data set, removes the volume serial number from the data set profile.

Erase-on-Scratch

Erase-on-scratch is an option that allows an installation to physically erase security-sensitive data at the time the data set extents are scratched (deleted or released for reuse). It permits a user to protect both single and multi-volume DASD data sets.

Erase-on-scratch when used together with a version of data management that supports this facility, ensures that when the data set is scratched it cannot be read by any program running under control of an IBM operating system. Installations have the option of controlling the erase attribute when creating and modifying a data set profile and when deleting actual data sets via the RACHECK and RACDEF preprocessing and post processing exit routines. (A reason code of 4 from the exit routines indicates that the data set is to be erased.) Options control how much erasure is performed. Note that if an installation activates ERASE ALL, an installation exit cannot override the option to prevent data sets from being erased.

When RACF determines that erase-on-scratch is to be performed, data management, not RACF, performs the actual data set erasure. Because erase-on-scratch places an additional load on DASD devices, it can have an impact on system performance depending on how much erasure is being performed.

Moving DASD Volumes Between Systems

When you move a DASD volume to a system that has RACF installed:

- If the DASD volume is defined on the new system, RACF performs authorization checking in the normal manner.

- If the DASD volume is not defined on the new system, or the DASDVOL class is not active, RACF protection or password protection is performed for individual data sets.

Tape Data Protection

This section provides information about RACF-protecting tape volumes and tape data sets. The following are considerations for using RACF to protect tape data:

- Tape data protection - discusses some aspects of tape data protection that a systems programmer may need to implement.
- Using utilities on RACF-protected tape volumes - lists the system utilities for which RACF performs authorization checking.
- Moving tape volumes between systems - discusses moving tape volumes and multivolume tape data sets from one system to another.

For more information on tape volume protection, see the *Security Administrator's Guide*.

Tape Data Protection and Bypass Label Processing (BLP)

You can use RACF to bypass label processing. If an installation specifies BLP at system generation or JES initialization and the user specifies BLP on the LABEL parameter of the DD statement, RACF issues a RACHECK to class FACILITY, resource ICHBLP, to provide additional control. To activate this additional control, installations must define the profile ICHBLP to RACF using the RDEFINE command. To add users and/or groups to the profile access list, an installation can use the PERMIT command.

If BLP is specified on the LABEL parameter and the system does not support BLP, the tape is treated as a nonlabeled (NL) tape.

Considerations for Nonstandard Label (NSL) Tapes

Nonstandard label (NSL) processing routines can provide RACF protection for tape volumes with nonstandard labels. You can provide protection by first defining the tape volumes to RACF (with the RDEFINE command or RACDEF macro instruction) and then issuing the RACHECK macro instruction in the NSL processing routine.

Before issuing the RACHECK and RACDEF macro instructions, the NSL routine should ensure that tape volume protection is active. Use the RACHECK macro instruction when a user requests access to a tape volume. Use the RACDEF macro instruction when PROTECT is specified on a JCL DD statement or a multivolume tape data set is extended to another volume (EOV processing). (The JFCBPROT indicator in the JFCB indicates that PROTECT has been specified.)

Considerations for Unlabeled (NL) Tapes

For a description of RACF considerations for opening non-labeled tapes for input and output, see *RACF Security Administrator's Guide.*

You should be careful when using non-specific volume requests for output volumes because the operating system assigns volume serial numbers and it is impossible for you to determine if the volume mounted is defined to RACF under a different number. If your installation plans to use RACF protection for non-labeled tapes, you should use JCL scans to prevent the use of non-specific volume requests and institute procedures to ensure that operators mount the correct tapes.

Considerations for MVS Tape Label Functions

The ACCLVL parameter on the RACHECK and RACDEF macro instruction specifies tape label access level information from the MVS tape label functions. The access value is passed to the RACHECK and RACDEF exit routines. YOu can also pass additional information to the exit routines with the ACCLVL operand.

Using Utilities on RACF-Protected Tape Volumes and Tape Data Sets

With the exception of IEHINITT, RACF performs authorization checking for tape volumes that are accessed by system utilities when these utilities issue the OPEN macro instruction; therefore, users of system utilities must be defined to RACF and have authorized access to any RACF-protected tape volumes that the utilities access.

The installation should restrict the use of the IEHINITT utility to only authorized administrators. You can use the RACF control program option to restrict the utility on an MVS/XA system.

Moving Tape Volumes Between Systems

When a tape volume is moved to a system that has RACF installed and the tape volume protection option is active, then:

- If the tape volume is defined in the RACF data set of the new system, RACF performs authorization checking in the normal manner.
- If the tape volume is not defined in the RACF data set of the new system, RACF performs authorization checking and the result is "not defined." Password checking is then done.

If the tape volume protection option is not active on the new system, RACF does not perform authorization checking.

Moving Multivolume Tape Data Sets Between Systems

When moving a multivolume tape data set that resides on a RACF-protected tape volume set, you might need to update the profile for the tape volume set on the destination system. This step is necessary if the multivolume data set was extended to additional volumes on one system and is then moved to another system where the tape volume set is defined to RACF. To update the profile for the tape volume set on the destination system, issue the RALTER command with the additional RACF-protected volumes specified on the ADDVOL operand.

RACF Data Set Considerations

This section describes the following considerations related to the installation's use of the RACF data set:

- Multiple RACF data sets - describes the use of multiple RACF data sets to reduce device contention and minimize the loss of a data set or device.

- Switchable RACF data sets - lists ways to provide a backup data set that you can switch to, without a re-IPL, if there is a failure on a primary RACF data set.

- Maintaining a RACF data set - describes how to deactivate the RACF data set so you can perform maintenance.

- Coordinating profile updates - describes how to update profiles so they are consistent and so that the updating does not interfere with other jobs in the system.

- Shared RACF data set considerations - describes what to do when the device that a RACF data set resides on changes from nonshared to shared and then back to nonshared without an intervening IPL.

RACF Data Set Volumes

The RACF data set must be a contiguous non-VSAM data set that resides on a DASD volume. At IPL time the data set is opened and allocated, and RACF control blocks are updated with the physical location of the data set on the volume. The data set is then closed. Because the control blocks are not updated again, you cannot move the RACF data set to another location on the same pack without causing I/O errors; therefore, all RACF data sets should be marked as unmoveable whenever DF/DSS DEFRAG is run against the volume. If the data set is moved, a re-IPL is required to access the RACF data set again.

Multiple RACF Data Sets

Multiple RACF data set support provides an alternative to maintaining all your RACF profiles on one data set. By using multiple RACF data sets (RACF allows up to 255), you can spread accesses across multiple devices. The use of multiple devices reduces device contention and the number of resources made unavailable by the loss of one data set or device.

Also see "Creating Multiple RACF Data Sets" in Chapter 1 and "RACF Data Set Storage Requirement" in Chapter 9.

Switchable RACF Data Sets

This facility allows an installation to identify alternate RACF data sets that can be switched to, without requiring a re-IPL, in case of a failure on a primary RACF data set. There is a one-to-one relationship between primary RACF data sets and backup data sets. None, some, or all of the primary RACF data sets can have corresponding backup data sets. RACF allocates the backup data sets at the same time as the primary data sets; therefore, the backup data sets must be online during RACF initialization. When electing to use this facility, the installation can choose from three levels of backup:

1. The backup data set is not updated as changes are made to the corresponding primary data set, but is available for switching when needed. Switching to this backup data set brings a down-level RACF data set into operation.

2. The backup data set is available for switching and is also updated to keep current with the primary data set. If this option is selected, the backup data set must be a copy of the corresponding primary data set existing at RACF initialization. Switching to this backup data set is transparent.

3. The third option is similar to the second except that changes made to the corresponding primary data set for the sole purpose of updating statistics counts are not reflected in the backup data set. Switching to this backup data set might cause the loss of some statistics data if you are maintaining statistics.

The cost, in terms of RACF processing, for option 1 is negligible, for option 2 is extremely high, and for option 3 is appreciable but not nearly as high as for option 2.

For more information on backup data sets, see "RACF Data Set Recovery" and "Failures During I/O Operations on the RACF Data Set" in Chapter 4 . The topic "Data Set Name Table" in Chapter 5 describes how the installation specifies the names of the backup data sets and the options to be used for each.

RACF provides several facilities for use in managing RACF data sets: the split/merge/extend utility (see Chapter 6, "RACF Utilities"), the RVARY and SETROPTS commands (see the *Command Language Reference*), and two tables - the data set name table (ICHRDSNT) and the range table (ICHRRNG) (see Chapter 5).

Maintaining a RACF Data Set

If a RACF data set requires maintenance because of error conditions in the data set, you can issue the RVARY command to deactivate the RACF data set and use the RACF data set verification utility program (ICHUT200) and the block update utility command (BLKUPD) for maintenance.

The following rules apply when **all** RACF data sets are deactivated by the RVARY command and RACF failsoft processing is active:

- On shared systems, a RACF data set is deactivated only on the system from which RVARY was issued.
- If a deactivated RACF data set is RACF-protected, the system operator must be aware that the system programmer will be using a utility to access the data set. Attempts to access RACF-protected resources will cause RACF to ask an operator to allow access to a resource. The RACHECK and RACDEF post-processing exit routines do not gain control when RACF failsoft processing is active.
- Batch and TSO users whose profiles are on a deactivated data set can enter the system as if RACF were not installed.
- Attempts to define resources to RACF with RACDEF SVC processing will causes an operator information message. The RACDEF request terminates with a return code of zero. Note that, because the request is allowed, the data set entries may be RACF-indicated but not be RACF-defined. After RACF is reactivated, use the ADDSD or RDEFINE commands to define the appropriate profiles from the information supplied in the operator messages.
- You cannot issue RACF commands against profiles on an inactivated data set.
- Commands that change multiple profiles will not perform all the operations if a required profile is on an inactive data set.

Note: Exit routines can examine the data set descriptor table created during RACF initialization to determine if a RACF data set has been deactivated by the RVARY command.

Coordinating Profile Updates

Installation personnel should plan to update profiles so that they remain consistent, and so that the updating does not interfere with other jobs running in the system.

Each individual operation performed by RACF serializes on a RACF data set, but a command or function may perform multiple operations on multiple profiles. For example, the CONNECT command changes both the user profile and the group profile. If two or more RACF commands or functions are executing at the same time and are making contradictory updates, their operations might be interleaved and, therefore, cause the information in the RACF data set to become incomplete or invalid.

Example: A DELUSER command is issued to delete a user from RACF. At the same time, the user, who has the ADSP attribute, is logged on the system and is creating a permanent user data set. (The data set will automatically be defined to RACF.)

The DELUSER command performs the following operations on the RACF data set:

1. Locates the user entry in the RACF data set.

2. Locates any user data set entries; ensures the user does not have any user data sets whose high-level qualifier is his userid. (RACF cannot delete the user until all his user data sets are deleted.)

3. Deletes the user entry.

4. Updates the group entry; removes the user as an eligible member of the group.

5. Deletes the connect entry for the user.

As a result of the ADSP attribute, RACF performs one operation on the RACF data set; RACF adds a data set entry for the permanent user data set.

In this example, if the user adds the new data set entry between Steps 2 and 3 of the DELUSER command, RACF adds a user data set entry to the RACF data set. However, RACF has already deleted the user who owns the entry. To avoid this situation, do not delete a user who is logged on the system or, to prevent a user from logging on, revoke a user (with the ALTUSER command) before deleting him.

Note: If a user is in the system and you update the user's attributes in the RACF data set using the ALTUSER or CONNECT command, the changes, except for the OWNER and AUTHORITY operands, do not take effect until the next time the user enters the system. However, a LISTUSER or LISTGRP command issued immediately after the change shows the new values.

Shared RACF Data Set Considerations

It is possible for the status of the device on which a RACF data set resides to change from nonshared to shared and then back to nonshared without an intervening IPL. This situation can occur if the device is defined as SHAREDUP at system generation time and the appropriate VARY CPU (for MVS/370) or CONFIG CPU (for MVS/XA) operator commands are issued.

In this case, the following restriction applies. When the device is shared by MVS systems, the processor (CPU) that processed the VARY or CONFIG command must do the RACF processing at least once before the status of the device is changed back to nonshared. To ensure that RACF processing takes place, have a RACF-defined user issue a TSO LOGON command or run a background job that contains the user parameter on the JOB statement. (This procedure is necessary because RACF needs to be aware of the changing of a RACF data set to shared status so that the resident index blocks and data blocks are maintained correctly.)

Other Considerations

This section describes:

- Multiple users per address space - for applications, such as IMS/VS, that permit multiple users per address space
- Default ACEE address in the TCB - for address spaces with multiple user, RACF allows a default ACEE pointed to by the TCB (task control block) under which the RACF request is being done
- Restarting jobs - for automatic or deferred restarts
- RACF and bypass password protection - for programs that execute with the bypass password protection bit set in the program properties table
- Removing RACF from your system - for removing RACF from your system when RACF is active
- Invoking the data security monitor - for users executing the data security monitor program

Multiple Users Per Address Space

When only one userid can be associated with an address space, user-related information (ACEE) is anchored from the address space extension block (ASXB). For applications that permit multiple users per address space, such as IMS/VS, each user requires an ACEE, and the application can request RACINIT CREATE to return a pointer to the ACEE rather than anchoring it from the ASXB. The application must then keep track of user/ACEE relationships by passing the appropriate ACEE pointer to RACHECK and RACINIT (CHANGE or DELETE), thus associating the correct ACEE with the user for whom processing is being done. RACINIT allows the invoker to specify a subpool from which the ACEEs and related storage can be obtained.

The program control option that provides protection for individual load modules will not provide protection for multiple users in an address space. If one user in a address space causes a program to be loaded, another user in the same address space can also execute the program.

Default ACEE Address in the TCB

RACF allows the issuer of the RACHECK, RACDEF, RACINIT, and RACLIST SVCs, and the FRACHECK service routine, to pass it the address of an ACEE. If no address is passed, RACF normally uses the ACEE for the address space, pointed to by the ASXBSENV field in the address space extension block (ASXB). Because most system service routines (for example, Open, Close, and EOV) do not accept the specification of an ACEE address when they are invoked, all data set opens in an address space are checked for authorization using the default ACEE. Therefore, in a multitasking environment, with many different users active in the same address space, all system service routine requests use the same ACEE, rather that the one that is specific to the user for whom the request is being done.

RACF allows the issuer of the RACHECK, RACDEF, RACINIT, and RACLIST SVCs and the FRACHECK service routine to pass it the address of an ACEE. If no ACEE address is passed, RACF uses a default ACEE, which is normally the address space ACEE pointed to by the ASXBSENV field in the address space extension block (ASXB).

However, because some callers of RACF do not pass an ACEE (for example the OPEN and EOV macros) and because some address spaces support multiple users, RACF allows each task (TCB) within an address space to have its own default ACEE pointed to by the TCBSENV field in the TCB extension block.

RACF handles requests as follows:

- RACINIT ENVIR=CREATE
  - If an address of a fullword to contain the ACEE address is passed to RACF, RACF stores the address of the newly created ACEE into the fullword pointed to by the given address.
  - If no address is passed, RACF stores the address of the newly created ACEE in the TCBSENV field. If the ASXBSENV field is presently set to binary zeroes, RACF also stores the new ACEE address in ASXBSENV. If the ASXBSENV field is nonzero, it is not modified.
- RACINIT ENVIR=DELETE
  - If an ACEE address is passed, the ACEE pointed to will be deleted.
  - If no ACEE address is passed, and the TCBSENV field is nonzero, the ACEE pointed to by TCBSENV will be deleted, and TCBSENV is set to zero. If TCBSENV and the ASXBSENV point to the same ACEE, then ASXBSENV is also set to zero.
  - If no ACEE address is passed and TCBSENV is zero, the ACEE pointed to by ASXBSENV is deleted and ASXBSENV is set to zero.
- RACHECK, RACDEF, RACLIST, and FRACHECK
  - If an ACEE address is passed, that ACEE is used.
  - If no ACEE address is passed and TCBSENV is nonzero, the ACEE pointed to by TCBSENV is used.
  - If no ACEE address is passed and TCBSENV is zero, then the ACEE pointed to by ASXBSENV is used.

When the RACINIT preprocessing exit routine sets a return code of 8, and when the RACINIT macro instruction request specified ENVIR=CREATE, the exit routine is responsible for creating and initializing the access control environment element (ACEE). When ICHRIX01 returns control, RACINIT attaches the ACEE passed back (either the original ACEE or the one created by the exit) to the ASXB and/or the TCB and copies the userid from the ACEEUSER field to the ASXBUSER field. (The exit should **not** modify these fields.)

Restarting Jobs

When a job automatically restarts and returns to a previous checkpoint, RACF repeats user verification and access authorization checking. If the job changed the password on the JOB statement, RACF uses the new password for user verification. But if the PASSWORD command or another job changes the password in the meantime, and JES user identification propagation is not used, RACF detects an invalid password and fails the job.

When you resubmit a job for a deferred restart, you should specify your current password on the JOB statement.

An additional consideration exists for tape volumes. If a user is restarting a job with a **deferred** step restart that has the PROTECT parameter on a DD statement, the job does not fail if the tape volume had been defined to RACF before the restart with the PROTECT=YES parameter. On the other hand, an **automatic** step restart with PROTECT specified is successful only if the step abnormally terminated before the tape data set was opened for output and before the tape volume was defined to RACF.

For either an automatic or deferred restart, RACF checks the current access authority at the time of the restart.

RACF and Bypass Password Protection

Programs that execute with bypass password protection (flag bit 6 is set in the program properties table IEFSDPPT) have the following effect on RACF processing:

- RACHECK macro instructions are bypassed by OPEN, EOV, checkpoint/restart, DADSM rename, and DADSM scratch. Thus, RACF does not perform authorization checking for RACF-protected data sets and tape volumes.

- RACDEF macro instructions are bypassed by OPEN, EOV, and DADSM scratch. Bypassing the RACDEF macro instructions can cause the following conditions:

  - After scratching a DASD data set protected by a discrete profile or performing tape label destruction (by OPEN or EOV) for a RACF-protected tape volume, RACF does not delete the RACF profile from the RACF data set.

  - A DASD data set with a discrete profile defined to RACF by DADSM allocate (as a result of ADSP or PROTECT) that extends to subsequent volumes will not have those volumes defined in the data set's discrete profile. You can later define these volumes using the ALTDSD command.

Removing RACF From Your System

You can use the following procedure to remove RACF. This procedure assumes that RACF is active (not deactivated by the RVARY command). The RACF commands used in the procedure must be issued by a user with the SPECIAL attribute. The user may also need to have the TSO MOUNT attribute.

1. Before removing RACF, you must prevent users from adding RACF-protection to data sets while you are deleting data set profiles from the RACF data set. To do this, you can revoke all users by using the ALTUSER command.

2. Remove RACF-protection from data sets in the following way:

   - Issue the SEARCH command (with the NOMASK operand) to locate all RACF-protected data sets. Request that a CLIST be created containing a DELDSD command for each data set name located in the scan.

     Note: Duplicate data set names require special handling because they require the VOLUME operand, but the SEARCH command does not put volume information in the CLIST.
   - Use the TSO EXEC command to execute the DELDSD commands in the CLIST data set. The DELDSD commands will remove each data set profile from the RACF data set and turn off the RACF indicator. For a DASD data set, the indicator is in the DSCB for a non-VSAM data set or in the catalog entry for a VSAM data set. For a tape data set, the indicator is in the TVTOC entry for the data set in the corresponding TAPEVOL profile. This step might involve mounting several volumes to complete the job.

3. Deactivate the RACF function in your system using the control information and instructions in the RACF program directory that you received from IBM. In general, the procedure consists of the following:

   - Use the procedure in the RACF program directory to update the libraries containing the RACF load modules.

   - Re-IPL to initialize the system without RACF.

   - Alter the MSTRJCL if necessary.

   - Uncatalog and scratch the RACF data set, if desired.

4. Remove the RACF operands from JCL, JOB, and DD statements, and notify users that they should do the same.

Using the Data Security Monitor

The data security monitor (DSMON) is a program that allows users with the RACF AUDITOR attribute or other users authorized through RACF to obtain a set of reports that provide information about the basic system integrity and security mechanisms for an installation. DSMON is an APF-authorized batch program that you normally run while RACF is active. If you run DSMON while RACF is inactive, DSMON produces only the system report.

A user who needs to run DSMON requires authorization to READ SYS1.PARMLIB unless DSMON is run on an MVS/XA system.

For more information about DSMON, see the *RACF Auditor's Guide*.

Chapter 3. Performance Considerations

The effect that RACF has on system performance depends directly on the type and number of RACF functions performed. The effect depends indirectly on the amount of overhead caused by page faults (during RACF processing) and by I/O contention for the RACF data set because of the need to serialize RACF processing.

This chapter identifies factors that affect system performance.

RACINIT Processing

When the TSO logon, IMS/VS or CICS/VS sign on, or batch job initiator issues the RACINIT SVC, RACF identifies and verifies the user to the system and performs terminal authorization checking. Therefore, system performance is affected by the frequency with which RACF-defined users log on to TSO, sign on to CICS/VS or IMS/VS, or submit batch jobs.

RACHECK Processing

When a resource manager issues the RACHECK SVC (such as during an OPEN), RACF performs authorization checking. An installation can choose to bypass normal RACHECK processing by using the global access checking facility. When global access checking allows a request, RACF does no I/O to the RACF data set, does no logging, and does not maintain any statistics. Global access checking provides a high performance path through authorization checking.

For systems with Always Call support, a global access table is recommended because of the increased frequency of RACHECKs that can occur. For systems where most users access their own data sets, you should include the entry &RACUID.*/ALTER in the global access table to bypass normal processing for a user's own data sets. (During normal processing, RACF always authorizes full access to a user's own data set.)

In addition, if generic profile checking is active during authorization checking, RACF builds lists of generic profiles in storage to be referenced repeatedly by the RACF SVCs, thus minimizing the frequency of I/O requests to the RACF data set. The use of generic profiles also decreases the size of the RACF data set and decreases the number of times you need to create and delete profiles.

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

FRACHECK Processing

FRACHECK uses the resident profiles constructed by RACLIST to perform authorization checking. FRACHECK performs no logging, gathers no statistics, issues no SVCs, and is branch entered by the resource manager.

The use of FRACHECK rather than RACHECK improves performance and is recommended for applications that have stringent performance requirements.

Notes:

1. *The resident profiles of FRACHECK are not refreshed when other profiles are refreshed by the SETROPTS command; the resource manager (for example, IMS/VS or CICS/VS) must refresh the profiles.*

2. *High use environments like IMS/CICS should place ICHSFR00, ICHRFR00, and ICHRFC00 in fixed LPA. See chapter 9 "Storage Estimates."*

RACDEF Processing

When DASD data sets are protected by discrete profiles (such as when a user has the ADSP attribute or specifies PROTECT=YES on a DD statement), the resource manager issues the the RACDEF SVC to add or delete resource profiles on the RACF data set. The frequency with which new resources are dynamically defined to RACF or RACF-protected resources are deleted affects the amount of RACF processing in the system. Therefore, you should use ADSP and PROTECT=YES only when discrete profiles are necessary.

Installation-Written Exit Routines

RACF provides preprocessing exits during RACF processing of:

- RACDEF, RACHECK, RACLIST, and RACINIT system SVCs

- FRACHECK

- ADDSD, ALTDSD, ALTUSER, DELDSD, DELGROUP, DELUSER, LISTDSD, PASSWORD, PERMIT, REMOVE, and SEARCH RACF commands

- ICHUT100 RACF utility

RACF provides postprocessing exits during RACF processing of:

- RACDEF, RACHECK, RACLIST and RACINIT system SVCs
- FRACHECK
- ADDSD, ALTDSD, DELDSD, LISTDSD, PERMIT, SEARCH, RLIST, RALTER and RDELETE RACF commands

RACF also provides an exit that is called by the RACF manager whenever it is necessary to store or compare encrypted password or OIDCARD data in a user profile.

Exit routines can add to or reduce the impact on system performance depending upon the processing the exit routines perform. See Chapter 7, "RACF Installation Exits" for details.

RACF also allows installations to create a naming convention table that RACF uses to check the data set names in all the commands and SVC routines that process data set names. Using the naming convention table degrades system performance slightly.

RACF Data Set

The placement of the RACF data set might affect system performance in the following ways:

Selection of Control Unit and Device: The RACF data set must be cataloged and should reside on a permanently resident volume. Placing the RACF data set on the same control unit or device as other frequently used data sets can degrade system and RACF performance.

Shared RACF Data Set:: RACF is designed so that its data set(s) can be shared between processor complexes while maintaining integrity. In installations where the RACF data set does not need to be shared, performance is optimized by placing the RACF data set on a non-shared device.

Choice of Device: The choice of the DASD device for the RACF data set can affect RACF and system performance in a high use environment. For example, device saturation occurs earlier for an IBM 3350 than it does for an IBM 3380.

If a cached auxiliary storage subsystem (such as IBM 3380 devices attached to an IBM 3880 Storage Control Model 13) is available, use it for the RACF data set. Also consider placing at least the first track of the RACF data set in the cache; this placement can have a significant effect on overall performance if the RACF data set is shared.

Multiple RACF Data Sets: The use of multiple RACF data sets can reduce the importance of the preceding factors because (1) fewer RACF requests might go to each RACF data set and (2) each RACF data set might be smaller, in which case each request would require fewer I/O requests and less arm movement.

Backup RACF Data Sets: Using backup RACF data sets and automatically keeping them current increases the amount of processing performed for updates to RACF data sets.

Logging

The AUDIT parameter on the ADDSD, ALTDSD, RALTER, and RDEFINE commands specifies when RACF is to log to the SMF data set the detected accesses and attempts to access RACF-protected resources. RACF restricts logging to the specified access type in a hierarchical manner. Thus ALL(READ) logs accesses at any level, but ALL(ALTER) logs accesses at the ALTER level only, thus decreasing the logging processing. The FAILURES(READ) option is the default. Selection of the ALL or SUCCESS(READ) option increases RACF and SMF processing while the NONE option eliminates all processing associated with logging.

The GLOBALAUDIT parameter on the ALTDSD and RALTER commands specifies which access attempts the user who has the AUDITOR attribute wants to log on the SMF data set. Regardless of the value specified on the GLOBALAUDIT keyword, the access attempts to be logged will not be less than those specified on the AUDIT keyword. Thus, selection of a value for GLOBALAUDIT that logs more access attempts than specified on the AUDIT keyword increases RACF processing.

Using the AUDIT parameter on the SETROPTS command and the UAUDIT parameter on the ALTUSER command specifies that RACF is to log all detected accesses to the RACF data set by RACF commands and the RACHECK and RACDEF SVCs issued by the audited user. Selection of these options increases RACF and SMF processing.

On the SETROPTS command, the auditor can select an option (NOSAUDIT) that commands issued by users with the SPECIAL or group-SPECIAL attribute are not to be logged and/or an option (NOCMDVIOL) that violations detected by RACF commands during RACF command processing are not to be logged. However, specifying UAUDIT or AUDIT on the SETROPTS command can increase RACF processing.

The auditor may also select the OPERAUDIT option on the SETROPTS command. The OPERAUDIT option allows the installation to audit all accesses to resources granted because the user has the OPERATIONS attribute or the group-OPERATIONS authority; however, specifying OPERAUDIT does increase RACF processing.

For more information on which activities are never logged, optionally logged, and always logged, see the *Auditor's Guide*.

Statistics Gathering

RACF records statistical information along with the descriptions of users, groups, and resources in the RACF data set. The NOSTATISTICS option on the SETROPTS command specifies that no resource statistics are to be recorded; NOINITSTATS specifies that no RACINIT statistics are to be recorded. Setting either or both of these options reduces RACF processing.

RACF Commands

During the processing of some RACF commands, a single RACF data set can be unavailable for other use (such as authorization checking). Any command that requires reading or processing a large number of profiles (such as, SEARCH, LISTDSD with the ID or PREFIX operands, LISTGRP *, LISTUSER *, and RLIST class-name *) can cause contention for the RACF data set because these commands repeatedly issue RESERVEs (one for each profile). Depending on the amount of contention, the processing of other RACF commands can be slowed down considerably, and other systems sharing the RACF data set can be locked out. To reduce the impact on the system, consider issuing RACF commands that perform large-scale operations against the RACF data set at slack time.

RACF Utility Programs

When one of the RACF utility programs is processing a single RACF data set, that RACF data set can be unavailable for other use (such as authorization checking). Therefore, to reduce the impact on the system and on RACF performance, run the RACF utility programs during slack time in system operation.

- The use of the BLKUPD utility command (ICHUT300) can cause long RESERVEs against the RACF data set because the RESERVEs are maintained from the time the terminal user issues the READ subcommand until the user issues the corresponding END command.

- The RACF data set verification utility program, ICHUT200, can obtain a working copy of the RACF data set defined by the SYSUT1 DD statement. If you use the copy function, ICHUT200 uses the copied data set and the RACF data set is available during subsequent processing of ICHUT200.

The use of resident index blocks or resident data blocks (data blocks include profile and BAM blocks as well as index blocks) enhances the performance of RACF processing by reducing the number of I/O requests made to the RACF data set. For each primary RACF data set, an installation can assign from 0 to 255 resident index blocks or data blocks to be maintained in the CSA.

If you have a data set name table (ICHRDSNT), you can specify the number of data blocks (or index blocks) for each primary RACF data set.

If you do not have a data set name table, you can either specify the number of index blocks for the single RACF data set in a field in the ICHSECOP module, or allow the default value of 10 resident index blocks to be used. (In ICHSECOP, you can specify only resident index blocks; you must use the data set name table to specify resident data blocks.)

You can also have resident index blocks or resident data blocks when the RACF data set resides on a shared device. In both cases, RACF updates the resident index or data blocks to ensure that all processors use the latest level of the blocks.

If you use the resident data block option (rather than having only resident index blocks), consider the following:

- All index blocks are eligible for residency, including the level one blocks.
- You should specify a larger number of blocks to allow as many of the frequently used index and profile blocks as possible to be kept in storage.

Note: When resident data blocks are used for a shared RACF data set, some resource statistics might not be updated. For more information, see "Selecting the Number of Resident Data Blocks" in Chapter 5.

Erase-on-Scratch

Because erase-on-scratch places an additional load on DASD devices, it can have an impact on system performance depending on how much erasure is being performed.

Also, failsoft processing used with erase-on-scratch can affect the erase-on-scratch procedure and overall system performance. If a RACHECK request with STATUS=ERASE is processed when RACF is inactive during failsoft processing and if the preprocessing exit grants authorization checking, then the reason code specified in the exit parameter list is passed to the caller of RACHECK. If the reason code equals 0, no erasure is performed. If the exit does not grant authority but failsoft processing does, then the reason code from the exit equals 4 and indicates erasure is to be performed. For how failsoft processing can affect system performance, see below.

Failsoft Processing

During failsoft processing, especially on an always-call system, the operator is prompted frequently; therefore, consider limiting the work performed on the system during failsoft processing.

List of Groups Processing

While list of groups processing is convenient in allowing certain users access, it does increase RACF processing for users connected to many groups.

Group Authority Processing

The user attributes for any user connected to a group with group-SPECIAL, group-AUDITOR, or group-OPERATIONS can be made to percolate for that user to subgroups owned by the group with one or more of these attributes. These subgroups can, in turn, own other subgroups also covered by the group attributes of the original owning group, and so on. Because the RACF data set must verify these group attributes among all groups and their subgroups, I/O processing can be high; therefore, you should limit percolation through the various groups and subgroups to what is actually required.

Chapter 4. Recovery Procedures

This chapter describes how to recover from a problem with the RACF data set by maintaining a backup copy of any primary RACF data set and using the RVARY TSO command to switch to the backup data set if the primary data set becomes unusable.

This chapter also describes the error recovery procedures that you can use when system or RACF failures occur during:

- I/O operations on the RACF data set
- RACF command processing
- RACF manager processing
- System operations on RACF-protected DASD data sets

Before attempting recovery from RACF failures, an installation should review the processing performed by any active RACF exit routines to determine whether the exits are obscuring the failures. For example, when command exits are being used to modify or eliminate the standard RACF naming convention, RACF error messages may specify qualifiers supplied by the exits rather than the high-level qualifiers of the data set names.

RACF, even when disabled in a system, provides protection and some services through failsoft processing.

Failsoft Processing

When all the RACF data sets in a system are inactive, failsoft processing provides protection and offers default services. RACF can use various internal tables to verify requests for protected resources, can route control to various exits for further processing, and can log the requests, whether they are granted or denied. (For more information on failsoft processing, see the *Security Administrator's Guide*.)

However, none of this additional processing occurs if there is no ACEE for the user. This means that if RACF is not active because of a failure during IPL, then failsoft processing prompts the operator. One reason for prompting the operator is that the user's identity may not have been verified (no password checking in RACINIT). This prompting also avoids the possibility of allowing accesses that would have been denied if, for example, an exit routine had been available. (The exit may be missing because of an error in ICHSEC00 before the exits are loaded.)

With always call, failsoft processing can cause operator prompts for almost every data set access in the system. Therefore, if the primary RACF data set is not usable, switching to the backup data set may be better than deactivating RACF.

RACHECK and RACDEF preprocessing exit routines are called during failsoft processing whenever RACF is active for that IPL and the RVARY command is issued to deactivate the RACF data sets. (Note: The postprocessing exit routines are not called.) These RACF exits allow an installation to define its own version of failsoft processing. Resource definitions may, for example, be recorded in SMF records and later automatically applied to the RACF data set by a user-written program. A resource access can be allowed, denied, or made to continue through normal failsoft processing.

RACF Data Set Recovery

The occurrence of a problem with the RACF data set is unlikely. Nevertheless, it is necessary to have recovery procedures in place should this ever happen.

RACF data set recovery consists of two parts:

- The dynamic maintenance of a backup copy of any "primary" RACF data set

- The RVARY TSO command, which you can use to switch to the backup data set if the primary data set becomes unusable. You can also set up a started procedure that invokes the TSO monitor program and issues an RVARY command if no TSO users are able to log on.

The backup data set must reflect the contents of the primary RACF data set. Naturally, the primary and backup should reside on different volumes. Once the installation has created the backup data set, RACF can maintain it automatically.

The majority of activity against the RACF data set is read only. Authorization checking, user verification, and the listing options all use only the primary data set. RACF makes changes to RACF profiles (like those resulting from ADSP or RACF commands), however, in both the primary and backup data sets. An installation can also specify the duplication of RACF statistics.

The RVARY command allows you to activate, deactivate, and switch RACF data sets without an IPL. You can also use RVARY to list the current configuration of RACF data sets. The SWITCH option of the command deactivates the current primary data set and then causes RACF to use the backup copy as the new primary. After a switch, the backup option is no longer in effect; you should reinstate it at the earliest opportunity.

Setup for RACF Backup

To create a backup RACF data set, you must copy the current RACF data set and specify the new data set configuration and backup options to RACF. No further updates to the primary data set should occur before the backup data set is activated. You should therefore create and activate backup data sets in an environment with no other users or jobs active in the system.

Creation of Backup Data Sets

The best way to create the backup data set is with the RACF split/merge/extend utility program (ICHUT400). For information on this utility, see “Chapter 6.”

When creating a backup RACF data set with ICHUT400, do not use the keyword LOCKINPUT. If you use LOCKINPUT, the primary (or source) RACF data set becomes unusable.

Definition of Dynamic Backup

The RACF data set name table (ICHRDSNT) specifies both the primary and backup RACF data set names and the recovery option. (If the primary data set is split, you specify several pairs of entries.) The following example shows the definition of one primary and one backup RACF data set.

```
ICHRDSNT  CSECT
          DC     AL1(1)             NUMBER OF ENTRIES
          DC     CL44‘SYS1.RACFP’   PRIMARY DATASET
          DC     CL44‘SYS1.RACFB’   BACKUP DATASET
          DC     AL1(10)            RESIDENT INDEX BLOCKS (see Note 1)
          DC     X‘80’              BACKUP OPTION FLAG (see Note 2)
```

Notes:

1. The default value is 10.
2. Flag Byte: B‘10.. ....’ duplicate updates except statistics
   B‘11.. ....’ duplicate updates including statistics
   B‘... ....1’ use the resident data block option on the primary data set.

Without the duplication of statistics, the activity against the backup data set tends to be insignificant. All read references are only to the primary data set and only changes affect the backup.

Because with statistics the level of activity increases sharply, you normally do not duplicate them. However, if you do not make regular copies on the backup data set and find it necessary to use the backup RACF data set copy, you risk having all inactive userids on the system revoked if you are using the SETROPTS INACTIVE option. To avoid such a possibility, you should use the ICHUT200 utility to make the copy from time to time. This utility updates the date of the last successful logon for the userids.

Additional Backup Measures

In addition to creating a backup data set, you should periodically take dumps of the RACF data set. The dumps could be part of the procedure to create backup copies of other important system data sets. If you create these copies with a dump/restore program while no other work is performed on the systems, the copy of the RACF data sets should be usable. If an inactive system cannot be guaranteed, you should use ICHUT200 or any other program issuing the proper ENQ/RESERVE for the RACF data set. ICHUT200 can only produce disk output; for tape, you should provide an additional copy step (using IEBGENER, for example). Of course, the RACF data sets and all copies should be RACF-protected at all times.

Along with this disaster recovery, you might find it necessary to do repair work on the system when RACF cannot be active. This repair work is possible only when the appropriate RACF exit routines or the operator, via failsoft, allow access to protected data sets. The exit routines should restrict these exceptional access rights to specific predefined TSO users and/or batch jobs.

The RVARY Command

The RVARY command is used for recovery when RACF fails. When you detect a problem with the RACF data set, you can enter the RVARY SWITCH command from any of the following:

- an active TSO session
- a batch job
- a started task that executes the TSO terminal monitor program.

When the RVARY SWITCH command is received, the console operator enters an installation-defined password to determine whether the change request is authorized and replies accordingly. A request can ask for RACF activation or deactivation or switching of the RACF data sets.

The RVARYPW operand on the SETROPTS command has two suboperands that allow a user with the SPECIAL attribute to define separate passwords. One suboperand defines a password to activate or deactivate RACF; the second operand defines a password to switch the RACF data set.

An installation can choose not to give the operator the passwords but rather to keep them under the control of the security administrator. The security administrator can then give the operator the passwords when needed. Once the operator receives any password, it should be changed for security purposes. If you choose not to provide password protection for RVARY, the operator must reply YES to allow RVARY to complete.

Caution The RVARY command is for recovery when RACF fails; therefore, RACF cannot control the use of the RVARY command.

A request to deactivate a current primary RACF data set also causes its backup to become inactive. You can deactivate the backup data set, however, while the corresponding primary remains active. A request to switch data sets causes the deactivation of the current primary data set and a shift of the primary function to the other backup data set (which RACF automatically activates, if necessary).

Multisystem Considerations

The use of the RVARY command becomes more complex in a multisystem environment. As a general rule, all systems must be synchronized with respect to the RACF data set configuration. To be synchronized, the RVARY command must be entered on all systems more or less simultaneously.

Under certain conditions, you might consider a different recovery approach. Assuming that the current primary RACF data set is only partially damaged and remains physically available, it is likely that the problem may be apparent on only one system. In that case, the best recovery strategy would be to restore the data set as soon as possible from that system without any manual interference on the other systems. To accomplish this restoration, you can copy the backup data set onto the primary data set (take a dump of the primary first) while updates are prohibited through ENQ/RESERVE. Unfortunately, you cannot use the VERIFY utility, because it issues ENQ/RESERVE only for its input data set; this application requires protection of the output data set. A user program developed for this purpose should use EXCP access to copy full tracks; tests have shown a significant performance improvement over block by block processing.

Synchronization Considerations

The occurrence of DASD errors may cause synchronization problems between the RACF data set and the contents of VTOCs and catalogs on a large or small scale.

Restoration of the RACF Data Set

If it becomes necessary to restore a RACF data set from tape, in most cases a resynchronization will be 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 data set would be useful in conjunction with manual checks.

Restoration of a Data Volume

When a single data volume has to be restored, the synchronization problem is limited. A manual procedure might be appropriate to correct the RACF definitions for that volume. Some installations use a TSO command procedure to analyze discrepancies.

Other Recovery Considerations

You can use RACF commands to accomplish all the synchronization steps (some may require the SPECIAL attribute). AMASPZAP and similar programs are unnecessary and should not be used. 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 data set before using RACF in production.

Failures During I/O Operations on the RACF Data Set

The effect of I/O failures on a RACF data set and the recovery procedures you use depend upon whether you use the data set name table (DSNT) to specify that a backup copy of a specific RACF data set is to be kept and whether it is to be kept current.

If you have repeated I/O errors on a RACF data set you must switch to another RACF data set. By specifying in the DSNT that a backup copy is to be maintained, you can use the SWITCH operand of the RVARY command to switch from a failing RACF data set to its backup without a re-IPL, whether the backup data set is being kept current or not. If you are not keeping a backup copy (specified in the DSNT) of the failing RACF data set, you must re-IPL.

Note: The data set name table is described in Chapter 5 under "Data Set Name Table."

If the failing RACF data set is shared between processor complexes, you must ensure that the failing RACF data set is not referenced on other systems after one system has switched to its backup. To stop access to the failing data set on all other systems, use the INACTIVE operand of the RVARY command on all other systems sharing the failing data set. If you have multiple RACF data sets, you will probably also use the DATASET operand of the RVARY command. You can issue RVARY with the SWITCH operand on the first system and then issue the same command on all other systems sharing the failing RACF data set. (For more information on the RVARY command, see the *Command Language Reference*)

At the end of this process, all the sharing systems will be using the backup data set. If this data set has been kept current (specified in the data set name table), it will be a logical copy of its primary RACF data set. If not, the data set will reflect the RACF structure at the time the data set was constructed.

After switching to a backup data set, you might analyze the failure in the old data set and either repair it or simply make another copy of the active data set. If the error is unrelated to the RACF data set organization and can be corrected (like a bad track for which an alternate can be assigned), you can correct the error. Use RVARY to inactivate the data set in use and then use ICHUT400 to make a copy of this data set on top of the repaired data set. If you are keeping backup data sets current, by switching to the new copy (using RVARY with the SWITCH and DATASET operands) you will again have a correct primary data set and a backup data set that is being kept current.

If you are keeping a backup copy that was created with the RACF data set verification utility program (ICHUT200), you must do one of the following:

- Follow the recovery procedures defined for your installation.
- Recreate any lost data with the block update (BLKUPD) utility command (ICHUT300).
- Re-IPL the system with RACF inactive. (Note that failsoft processing is in effect.) You can then run the ICHUT200 utility program to identify inconsistencies in the internal organization of the RACF data set. Then use the BLKUPD utility command (ICHUT300) to fix the RACF data set.
- Re-IPL the system using the backup RACF data set.

  Ensure that all necessary updates are made to bring it up to the correct level. You can use the NOSET operand on the ADDSD or DELDSD command if there are discrepancies between the data set profiles in the RACF data set and in the backup copy. **Note**: The backup copy is the one you created using the RACF backup utility program (ICHUT200).

If you have not defined a backup data set in the data set name table (DSNT), consider the following:

- Maintain a backup copy of the RACF data set on a periodic basis. (You can use the RACF utility ICHUT200 to create a backup copy.)

- Log all changes to the RACF data set by specifying the AUDIT parameter with the * operand on the SETROPTS command.

You can then partially recreate the RACF data set by using the back-up copy and updating it with the detected changes to the RACF data set that are identified in the SMF type 80 records (RACF processing records).

Failures During RACF Command Processing

System or RACF failures that occur during the processing of RACF commands can cause discrepancies between the profiles on the RACF data set. (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 userid.)

In this section, the RACF commands are grouped in categories based on the operations performed by the commands on the RACF data set. It should be noted that the use of some command operands, such as NOSET, requires the command issuer to have a specific authority in order to be effective.

Commands That Do Not Modify RACF Profiles

The commands that do not modify RACF profiles are:

- LISTGRP
- LISTDSD
- RLIST
- LISTUSER
- RVARY
- SEARCH
- SETROPTS

Failures that occur during the processing of these commands do not cause problems with the profiles on the RACF data set because these commands do not modify profiles. However, the SETROPTS command does rewrite the index control block (ICB) in the "master" RACF data set.

To recover, perform the following steps:

1. Examine the error messages to identify the failure.
2. Reissue 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 data set. 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. Reissue 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, reissue the command.

4. If the user or group profiles have discrepancies, issue the appropriate commands to correct the data in the profiles.

   Example: A failure occurs during the processing of the ADDUSER command and the user profile is created correctly but the group profile is not updated with the new user's userid. In this case, issue the CONNECT command with the default group name as the desired group in order to update the group profile.

5. If there are no discrepancies and the user and group profiles are correct, the command completed successfully.

6. 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 data set. 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, reissue 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 data set. Therefore, failures that occur during the processing of these commands can cause discrepancies between the profiles on the RACF data set, or discrepancies between data set profiles and the RACF-protected indication for the data set.

The commands are:

- ADDSD
- ALTDSD (with the ADDVOL, ALTVOL, or DELVOL operand)
- 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, and data set 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, issue the appropriate commands to correct the profiles.

   Example 1: 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, issue the ADDSD command with the NOSET operand to create the data set profile.

   Example 2: 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, issue the DELDSD command with the NOSET operand.

   Example 3: 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 userid from the group profile. In this case, reissue the REMOVE command.

5. If the failure occurs again, contact your programming support representative.

Failures During RACF Manager Processing

The RACF manager performs operations on the RACF data set(s) 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 data set.

For messages ICH402I, ICH403I, and ICH404I, see *RACF Messages and Codes* for the error recovery procedures listed with each message under the heading "Problem Determination." Also see the *RACF Program Logic Manual* for information about diagnosing problems with the RACF data set.

For messages other than ICH402I, ICH403I, and ICH404I that indicate a failure occurred during RACF manager processing, the system programmer or security administrator performs the following steps:

1. Reissue the RACF command or RACF utility, or perform the system operation again.

2. If the failure occurs again, then:

   a. Run the RACF data set verification utility program (ICHUT200) to identify the problem with the RACF data set.

   b. Use the block update (BLKUPD) utility command (ICHUT300) to correct the problem in the RACF data set.

   c. Rerun the ICHUT200 utility program to determine if there are any additional problems. If so, use the BLKUPD utility command to correct the additional problems.

For messages ICH402I, ICH403I, and ICH404I, the system programmer or security administrator should perform steps 2a and 2b.

See the *RACF Program Logic Manual* for detailed information about the RACF manager and the RACF data set.

Failures During System Operations on RACF-Protected Data Sets

Failures during system operations affect only data sets that are protected by the use of discrete profiles. (These system operations do not not automatically create modify, or delete generic profiles.)

System failures that occur during the processing of the following operations can cause discrepancies between the data set profiles in the RACF data set and the indication that a data set is RACF-protected in the DSCB (for non-VSAM data sets) or the catalog (for VSAM data sets).

- SCRATCH (non-VSAM) and DELETE (VSAM)
- ALLOCATE (non-VSAM) and DEFINE (VSAM)
- RENAME (non-VSAM) and ALTER (VSAM)
- EOV (non-VSAM)

The recovery procedures are similar for VSAM and non-VSAM operations.

Failures During SCRATCH or DELETE

System failures that occur when scratching (or deleting) RACF-protected DASD data sets can cause the RACF indicator to be left on but delete the data set profile in the RACF data set.

To recover, perform the following steps:

1. Examine the error messages to identify the failure.

2. List the data set profile for the affected data set.

3. If the volume information in the data set profile still exists, rerun the SCRATCH (or DELETE) operation.

4. If the volume information in the data set profile does not exist, use the TSO LISTDS (or access method services LISTCAT) command to determine if the RACF indicator is set in the DSCB (or catalog entry) for the data set.

5. If the RACF indicator is still set, issue the ALTDSD command with the NOSET and ADDVOL operands to recreate the information in the data set profile. If the data set profile does not exist, issue the ADDSD command with the NOSET operand to recreate it. Then rerun the SCRATCH (or DELETE) operation.

Failures During ALLOCATE or DEFINE

Failures that occur when allocating (or defining) RACF-protected DASD data sets can cause the data set profile to be created without setting the RACF indicator in the DSCB (or catalog).

To recover, perform the following steps:

1. Examine the error messages to identify the failure.

2. List the data set profile.

3. If the profile does not exist, rerun the ALLOCATE (or DEFINE) operation.

4. If a profile exists, use the TSO LISTDS (or access method services LISTCAT) command to determine if the RACF indicator is set in the DSCB (or catalog entry) for the data set.

5. If the RACF indicator is not set, perform the following steps:

   a. Issue the DELDSD command with the NOSET operand to delete the data set profile.

   b. Rerun the ALLOCATE (or DEFINE) operation.

Failures During RENAME or ALTER

Failures that occur when renaming a RACF-protected DASD data set can cause discrepancies between the name in the data set profile and the name in the DSCB (or catalog entry).

To recover, perform the following steps:

1. Examine the error messages to identify the failure.

2. List the data set profile.

3. If the data set profile has not been updated with the new name, rerun the RENAME (or ALTER) operation.

4. If the name in the data set profile has been updated but the name in the DSCB (or catalog entry) has not been updated, then perform the following steps:

   a. Issue the ADDSD command with the NOSET operand and use the old data set name.

   b. Issue the PERMIT command with the FROM operand using the new data set name.

c. Issue the DELDSD command with the NOSET operand and use the new data set name.

d. Rerun the RENAME (or ALTER) operation.

Failures During EOV (Non-VSAM)

Failures that occur during end-of-volume (EOV) processing can cause discrepancies between the DASD data set profile and the RACF indicator in the DSCB for that volume.

To recover, perform the following steps:

1. Examine the error messages to identify the failure.

2. List the data set profile.

3. If no change has been made to the data set profile, rerun the step or job containing the EOV operation.

4. If the data set profile has been updated with the volume, use the TSO LISTDS command to determine if the RACF indicator is set in the DSCB for the volume.

5. If the RACF indicator for the volume is not set, perform the following steps:

   a. Issue the ALTDSD command with the NOSET and DELVOL operands.
   b. Rerun the step or job containing the EOV operation.

Chapter 5. RACF Options

This chapter separates the options available to RACF into the following categories:

- Specifying RACF data set options - includes the data set name table, range table, and ICHSECOP module

- Specifying resource class options - includes a description of resource classes, the class descriptor table, system authorization facility and MVS router, RACF router table, and SETROPTS command

- Using started procedures - associates the names of started procedures with RACF userids and group names

- Using the data encryption standard (DES) algorithm - encrypts passwords and OIDCARD data before they are stored in the RACF data set

- Changing the ICHAUTAB module - contains the RACF authorized-caller table

- Changing the ICHRSMFI module - contains the default values for the SORT/MERGE parameters, the dynamic allocation parameters, and the processing options used by the RACF report writer

Specifying RACF Data Set Options

You can specify options for the RACF data set using the:

- Data set name table - describes the RACF data sets to RACF and allows you to select the number of resident index/data blocks

- Range table - determines the RACF data set to be accessed for a particular profile

- ICHSECOP module - allows you to bypass RACF initialization processing (RACF is inactive), select the number of resident index blocks, and disallow duplicate names for discrete data set profiles.

Data Set Name Table

The data set name table (ICHRDSNT) is an installation-written load module that describes the RACF data sets to RACF. This table may reside in SYS1.LINKLIB or any other library concatenated to SYS1.LINKLIB via the LNKLSTxx member in SYS1.PARMLIB. This table contains entries describing each RACF data set and its backup data set. A data set's position in this table corresponds to the data set number in the range table. The first byte of the table is a binary number indicating the number of entries in the table. Each entry consists of two 44-byte data set names, a 1-byte resident index block (or data block) count field, and a 1-byte flag field.

The first data set name is that of a primary data set. If this field is an asterisk, RACF prompts the operator during RACF initialization, to supply the data set name. The second data set name is that of an associated backup data set. If this field is an asterisk, RACF prompts the operator again. A blank data set name field indicates the absence of either a primary or backup data set for this RACF data set. The first primary RACF data set is considered to be the "master" RACF data set.

The resident index block count field specifies the maximum number of index blocks to be kept resident for the primary data set while it is active. If you select the resident data blocks option, this field specifies the maximum number of index, BAM, and profile blocks to be kept resident.

The format of the flag field is as follows:

Bit Setting	Meaning
00..	No updates are to be duplicated on the backup data set
10..	All updates, other than statistics, are to be duplicated on the backup data set
11..	All updates, including statistics, are to be duplicated on the backup data set
....1	Use the resident data block option for the primary data set

If no table is provided, RACF is initialized with a single RACF data set.

The following example shows how the data set name table can be used. Assume that an installation arranges its data set profiles in the following manner: test data set profiles on SYS1.RACFDS1, production data set profiles on SYS1.RACFDS2, and system data set profiles on SYS1.RACFDS3. For recovery, the installation wants a backup data set for each primary RACF data set. In addition, except for SYS1.RACFDS1, the installation wants all updates to the primary RACF data sets duplicated in its corresponding backup data set. For SYS1.RACFDS1, the installation wants all updates other than statistic updates duplicated in the backup data set. SYS1.RACFDS1 and SYS1.RACFDS2 use resident data blocks, instead of just resident index blocks.

The following data set name table correctly follows this criteria:

AL1(3)	Number of primary RACF data sets
CL44'SYS1.RACFDS1'	Name of first RACF data set (test data sets)
CL44'SYS1.BACKUP1'	Name of first RACF data set backup
AL1(20)	Number of resident blocks
XL1'81'	Flags - all updates other than statistic updates are to be duplicated in the backup data set. The resident index blocks include both index and non-index blocks.
CL44'SYS1.RACFDS2'	Name of second RACF data set (production data sets)
CL44'SYS1.BACKUP2'	Name of second RACF data set backup
AL1(10)	Number of resident blocks
XL1'C1'	Flags - all updates, including statistics, are to be duplicated in the backup data set. The resident index blocks include both index and non-index blocks.
CL44'SYS1.RACFDS3'	Name of third RACF data set (system data sets)
CL44'SYS1.BACKUP3'	Name of third RACF data set backup
AL1(10)	Number of resident index blocks
XL1'C0'	Flags - all updates, including statistics, are to be duplicated in the backup data set.

Selecting the Number of Resident Index Blocks

You can select the maximum number of RACF data set index blocks to be made resident in the common service area (CSA). An installation can specify from 0 to 255 resident index blocks per primary RACF data set. The default value is 10 resident index blocks.

Resident index blocks reduce the amount of I/O that is required to service the RACF data set. Each index block uses 1056 (1K + 32) bytes of storage in the CSA.

During IPL, RACF loads as many index blocks as will fit, starting with the highest-level index blocks. If RACF does not use all the allocated buffers, it frees the unused buffers. Once they are loaded, the index blocks remain in their buffers; if a new index block is created during RACF operation, the block does not become resident until the next IPL.

Note that when the RACF data set resides on a shared device, the level one index blocks cannot be resident; therefore, the value you specify refers to level two and higher index blocks.

Selecting the Number of Resident Data Blocks

In the data set name table (ICHRDSNT), you can specify the number of resident data blocks for each primary RACF data set. This option (setting bit 7 in the flag field) will keep any type of data block (profile and BAM blocks as well as index blocks) resident in the common service area (CSA). An installation can specify from 0 to 255 resident data blocks; the default value is 10 resident data blocks.

Resident data blocks reduce the amount of I/O that is required to service the RACF data set. This function is also dynamic because, at any time, only the most frequently used blocks are kept in storage. Each data block uses 1056 (1K + 32) bytes of storage in the CSA.

At RACF initialization time, ICHSEC00 obtains the number of buffers specified in the data set name table. The RACF manager then keeps track of when each buffer was used last.

For a nonshared RACF data set, the RACF manager does the following processing:

- 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 an old 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 data set has an up-to-date version of all blocks.

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 data set system.) If it finds none, the manager takes the buffer containing the least-recently used block.

For a shared RACF data set, the processing is different from a nonshared RACF data set in the following ways:

- 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.

- ALTERI requests do not cause the ICB to be rewritten. Therefore, the other processors (CPUs) will not know that statistics information has changed; thus, some statistics information may be lost.

Notes:

1. *If the RACF data set is shared, you do not have to specify the resident data block option for all processors (CPUs) that share the RACF data set. However, if you specify the resident data block option for any of these processors, all of the processors must be using RACF Version 1 Release 6 or later.*
2. *You can specify resident data blocks only with the data set name table; ICHSECOP can specify only resident index blocks.*
3. *When you select this option, all index blocks are eligible for residency, including level one blocks.*
4. *The default is not to activate this option. The resident data block option, if selected, replaces the resident index block option. If the current number of in-storage blocks is optimized for resident index blocks, you should specify a larger number of resident blocks to reduce the amount of I/O. If you do not specify a larger number of blocks, index operations may experience a performance degradation because some of the resident blocks are now used for other types of data.*

Range Table

The range table (ICHRRNG) is an installation-modifiable load module that determines which profiles RACF places in each the RACF data set. This table must reside in a link pack area (LPA) library or modifiable link pack area (MLPA) library, such as SYS1.LPALIB. The format of the IBM-supplied table (which defaults to a single RACF data set unless modified) is as follows:

F'1'	Number of range values
XL44'000----000'	Range start value
AL1(1)	Data set number

You must arrange the table in ascending order of the 44-byte strings. The first range start value must contain 44 bytes of binary zeroes. The 8-bit binary data set number represents a RACF data set number from 0 to 255 indicating the data set's relative position in the data set name table (ICHRDSNT). If zero is specified for this number, it indicates that the range is not associated with a data set. The RACF manager returns a code of 28 when an attempt is made to access an entity in such a range. If the 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 data set 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.

RACF handles the entries in general resource classes in a special way. RACF alters the names of entries in these classes internally to allow entries from *different* classes to have externally identical names. The alteration consists of prefixing five characters, the first four characters of the class name plus a dash, to the beginning of the entry name. For example, a TAPEVOL named DATA1 becomes TAPE-DATA1 and a DASDVOL named DATA1 becomes DASD-DATA1.

RACF modifies generic profile names internally as follows:

- For DATASET profiles, the first delimiter (a period) is converted to X'01'. Each percent (%) is converted to X'FB'. Each asterisk (*) is converted to X'FC'. If the last character in the name is an asterisk, X'FE' is appended to the end.

- For general resource classes, the hyphen (-) that is added internally is converted to X'02'. Each percent (%) is converted to X'FB'. The asterisk is converted to X'FC'.

Names of CONNECT profiles consist of user-name and group-name, separated by X'00'.

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

The following example shows how the range table can be used:

An installation wants all profile names beginning with "GRPA" through "GRPF" and "TEST1" through "TEST8" to reside on data set 2, all profile names beginning with "SYS1" to reside on data set 3, and all the remaining data to reside on data set 1. The following range table correctly follows these criteria:

```
F'7'                              Number of range values
XL44'000------000'                Range start
AL1(1)                            Data set number
XL44'C7D9D7C1000--000'*           Range start GRPA
AL1(2)                            Data set number
XL44'C7D9D7C7000--000'            Range start GRPG
AL1(1)                            Data set number
XL44'E2E8E2F1000--000'            Range start SYS1
AL1(3)                            Data set number
XL44'E2E8E2F2000--000'            Range start SYS2
AL1(1)                            Data set number
XL44'E3C5E2E3F1000--000'          Range start TEST1
AL1(2)                            Data set number
XL44'E3C5E2E3F9000--000'          Range start TEST9
AL1(1)                            Data set number
```

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

Selecting Options with ICHSECOP

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 index blocks
- 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 (RACF is activated), ten index 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 five bytes of data formatted as follows:

Byte 0	Bit 0	Bypass RACF initialization processing (when set on).
	Bit 1	Disallow duplicate name for data set profiles (when set on).
	Bits 2-7	Reserved.
Bytes 1-4		The number of index blocks to be made resident.

Bypassing RACF Initialization Processing

If you want to make RACF inactive, you can bypass RACF initialization processing during IPL by setting bit 0 of byte 0 on in module ICHSECOP. You can use this option as part of the procedure for bypassing RACF functions any time after the installation of RACF is complete.

This option (setting bit 0 on) makes RACF inactive until you turn bit 0 off and re-IPL. If this option is in effect (RACF is inactive), you cannot use the RVARY command to make RACF active.

When this option is used, a RACINIT SVC is not issued 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 RACHECK SVC is still issued. If you are using any RACF-protected resources on your system:

- Use the SETROPTS command to turn off resource protection prior to bypassing RACF initialization processing

- 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 except when 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 Index Blocks

You can select the number of RACF data set index blocks to be made resident in the common service area (CSA). An installation can allocate from 0 to 255 resident index blocks; the default value is 10 resident index blocks.

Resident index blocks reduce the amount of I/O that is required to service the RACF data set. Each index block uses 1056 (1K + 32) bytes of storage in the CSA.

If your installation has a data set name table (ICHRDSNT), you can specify the number of resident index blocks for each primary RACF data set by using the data set name table. (See "Data Set Name Table" in this chapter.)

If your installation does not have a data set name table, you can specify the number of resident index blocks for a single RACF data set in ICHSECOP, or accept the default value of 10 resident index blocks.

Note that when the RACF data set resides on a shared device, the level one index blocks cannot be resident; therefore, the value you specify in module ICHSECOP refers to level two and higher index blocks.

Disallowing Duplicate Names for Data Set Profiles

RACF provides an option that disallows identically-named data sets to be defined to RACF with separate, discrete profiles. When you use this option, the RACF manager disallows the ADDSD command or creation of **new** data sets via ADSP or PROTECT=YES, if a discrete data set profile with the same name already exists on the RACF data set.

To disallow profiles with the same data set name, you must set bit 1 of byte 0 on in module ICHSECOP.

Note: For RACF classes other than DATASET, you can never have duplicate profile names defined to RACF within the same class.

Specifying Resource Class Options

The resources that RACF can protect are divided into two categories: data sets and general resources. Classes of general resources are defined in a class descriptor table and include DASD volumes, tape volumes, terminals, applications, and IMS/VS and CICS/VS transactions. This section describes considerations related to using RACF with these resources.

General Resource Classes

With the exception of the DATASET, USER, and GROUP classes, all resource classes are represented in the class descriptor table (CDT). The CDT consists of two modules: ICHRRCDX is for IBM-supplied entries; ICHRRCDE is for installation-defined entries. The IBM-supplied CDT contains class descriptors for the following classes:

- DASDVOL (DASD volumes)
- TAPEVOL (tape volumes)
- TERMINAL (terminals)
- TIMS (IMS/VS transactions)
- GIMS (IMS/VS transactions group)
- AIMS (IMS/VS application group names)
- APPL (applications)
- TCICSTRN (CICS/VS transactions)
- GCICSTRN (CICS/VS transactions group)
- PCICSPSB (CICS/VS program specification blocks or PSBs)
- QCICSPSB (CICS/VS PSBs group)
- GMBR (global access checking group)
- GLOBAL (for global access checking)
- DSNR (DB2)
- FACILITY (bypass label processing)
- SCDMBR (security classification of users and data member)
- SECDATA (for security classification of users and data)
- FCICSFCT (file control table)
- HCICSFCT (file control table group)
- JCICSJCT (journal control table)
- KCICSJCT (journal control table group)
- DCICSDCT (destination control table)
- ECICSDCT (destination control table group)
- SCICSTST (temporary storage table)
- UCICSTST (temporary storage table group)
- MCICSPPT (processing program table)
- NCICSPPT (processing program table group)
- ACICSPCT (program control table)
- BCICSPCT (program control table group)
- PROGRAM (for programs)
- PMBR (program member)
- VMMDISK (VM minidisks)
- VMNODE (VM nodes)
- VMRDR (VM readers)
- VMBATCH (VM batch jobs)

RACF commands and SVCs reference the CDT whenever a class name other than DATASET, USER, or GROUP is received as input. Each IBM class descriptor is a CSECT in load module ICHRRCDX. The last CSECT in the load module is ICHRRCDE, which indicates the end of the descriptor table.

Note: The class descriptor table must be the same for each system sharing the RACF data set. If they are not the same, you might get unpredictable results when using the SETROPTS command to activate or deactivate RACF.

Defining New Classes to RACF

An installation can, as needed, add new class descriptors or modify or delete old class descriptors in the CDT for installation-supplied entries (ICHERCDE). You use one of the following methods:

- Invoke the class descriptor macro ICHERCDE for each resource class and do an assemble, then link edit the result to produce the load module used by RACF. When you use this method, the ICHRRCDE macro cross-checks class descriptors to ensure that no errors exist (for example, that the first four characters of class names are unique). If you are installing RACF for the first time, use this method.

- Link edit the object module(s) for the class(es) you are adding or modifying together with the existing load module ICHRRCDE in SYS1.LINKLIB to produce a new load module. Be sure that the last CSECT in the load module is ICHRRCDE. You can delete a class from the descriptor list by specifying the name of the class to be deleted using the linkage editor REPLACE statement. (Do not delete or modify any IBM-supplied classes from the load module, ICHRRCDX.) When you use this method, you can add object modules for new classes to the load module without reassembling. If you have made changes after installing RACF, use this method. However, if you do not completely reassemble the class descriptor table, you lose the cross checking that the ICHERCDE macro performs.

Defining the Class Descriptor Table

The ICHERCDE macro generates entries for the resource class descriptor table. The class descriptor table contains information that directs the processing of general resources. The table consists of two load modules, ICHRRCDX, which contains IBM-defined entries and ICHRRCDE, which contains installation defined entries. The table has an entry for each class except the USER, GROUP and DATASET classes. To generate the table, you must specify the macro once for each class. To identify the end of the class descriptor table, you invoke the macro without specifying any operands.

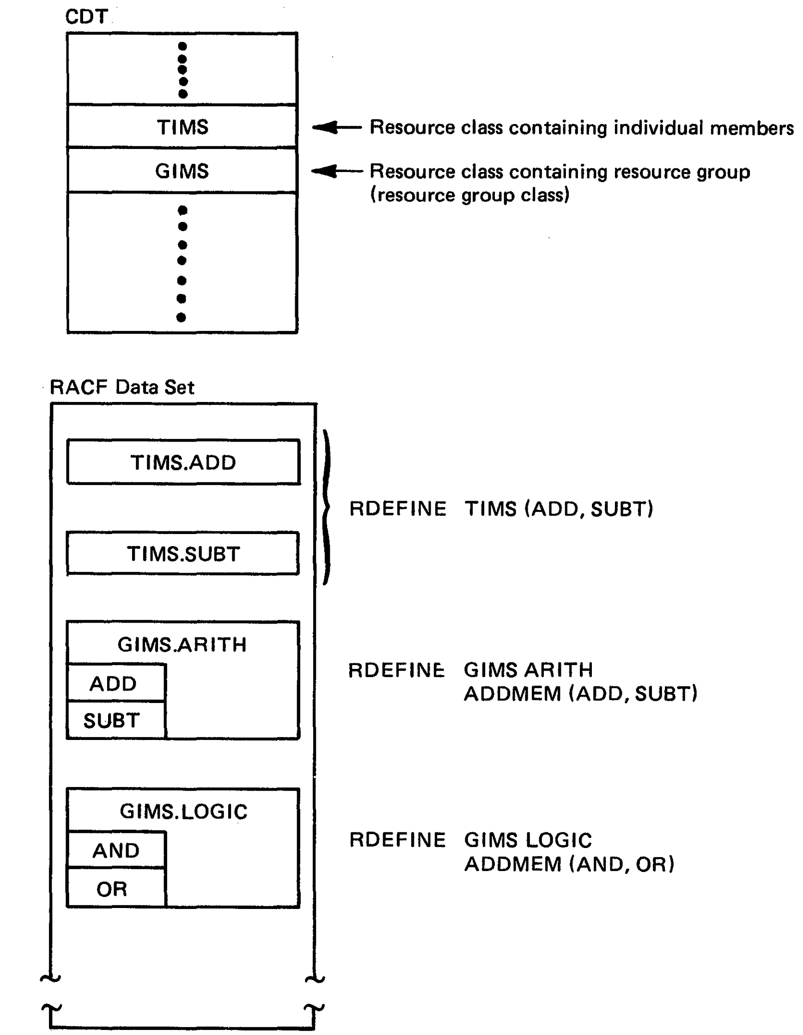
Note: If an entry added to the class descriptor table is to be accessed by the RACROUTE macro instruction, you should also code an ICHRFRTB macro instruction for the entry. See "RACF Router Table" later in this chapter.

The ICHERCDE macro produces a CSECT for each invocation. If the CLASS operand is present, the CSECT name is the name of the class being defined; otherwise, the CSECT name is ICHRRCDE.

For information on coding the ICHERCDE macro, see "ICHERCDE Macro" in Chapter 10.

Resource Groups

Resource groups protect, with one RACF profile, a set of resources that have the same security requirements (for example, a given user or group has the same authority to all the resources in the group). Figure 5-1 provides a sample of the creation of resource groups and resource group classes.


CDT
TIMS
GIMS
Resource class containing individual members
Resource class containing resource group (resource group class)
RACF Data Set
TIMS.ADD
TIMS.SUBT
RDEFINE TIMS (ADD, SUBT)
GIMS.ARITH
ADD
SUBT
RDEFINE GIMS ARITH ADDMEM (ADD, SUBT)
GIMS.LOGIC
AND
OR
RDEFINE GIMS LOGIC ADDMEM (AND, OR)

Notes:

- ADD and SUBT are defined to RACF as members of the resource class TIMS.
- The resource group ARITH is defined to RACF and the ADDMEM operand is used to add the members ADD and SUBT that also have resource profiles in the TIMS class.
- The resource group LOGIC is defined to RACF and the ADDMEM operand is used to add the members AND and OR to the group profile. No individual resource profiles need to be defined for AND or OR.

Figure 5-1. Creating Resource Groups

To make use of resource groups for a given class of resources (for example, TIMS), the installation defines a resource group class (in this example, GIMS) associated with the given class of resources. To group a set of resources in the given class, the installation defines an entity in the associated resource group class and makes each resource in the set a member of the grouping entity. When a user is allowed to access the grouping entity (with the PERMIT command), RACF propagates the authority to each member resource. RACF also propagates the universal access of the grouping entity to each resource.

When planning for the use of resource groups, consider the following:

- Resource groups are effective only when used in combination with the RACLIST facility. RACF does not automatically propagate the access list information and universal access of a grouping entity to its member resources when the resources are made members of the grouping entity or when users or groups are allowed access to the grouping entity with the PERMIT command. Instead, this propagation takes place when the RACLIST facility is used to construct in-storage profiles for a given resource class. Resource grouping is possible, therefore, only for those classes of resources for which the resource manager (for example, IMS) invokes the RACLIST facility prior to performing authorization checking.

- A grouping entity is a RACF-protected resource. Thus, RACF controls access to the entity and any user accessing that entity must have sufficient authority to perform the desired operation. A resource grouping class is associated with one and only one resource class and cannot be used to group resources from two different classes; nor can a class be grouped by two different resource group classes.

- The resource group classes cannot include generic profile definitions, but a grouping entity can contain generic names as members.

System Authorization Facility (SAF)

The system authorization facility (SAF) provides a system that gets control in response to a request from a resource manager. SAF conditionally directs control to RACF, if RACF and/or an installation-supplied processing routine is present. SAF does not require any other program product as a prerequisite, but overall system security functions are greatly enhanced and complemented by the concurrent use of RACF. The key element in SAF is the MVS router.

MVS Router

SAF provides an installation with centralized control over system security processing by using a system service called the MVS router. The MVS router provides a focal point and a common system interface for all products providing resource control. The resource managing components and subsystems call the MVS router as part of certain decision-making functions in their processing, such as access control checking and authorization-related checking. These functions are called "control points." This single SAF interface encourages the use of common control functions shared across products and across systems.

The router is always present whether or not RACF is present. If RACF is available in the system, the router passes control to the RACF router (ICHRFR00) that invokes the appropriate RACF function based on the parameter information and the RACF router table (ICHRFR0X or ICHRFR01), which associates router invocations with RACF functions. For more information on the MVS Router, see "MVS Router" in Chapter 7.

Router Table Entry Definition

The RACF router table consists of two parts: ICHRFR01, the installation-defined module, and ICHRFR0X, the IBM-supplied module. You can use the ICHRFRTB macro to generate entries in the optional installation-defined part of the RACF router table, ICHRFR01. This table controls the action taken by the RACF router, ICHRFR00, when it is invoked by the RACROUTE macro instruction.

The IBM-supplied module ICHRFR0X contains one entry for each entry in the class descriptor table, plus one entry each for DATASET and USER. For all entries, the operands REQSTOR and SUBSYS have the default value (all blanks), and the ACTION operand is set to RACF.

Note: If an entry added to the class descriptor table is to be accessed by the RACROUTE macro instruction, you should also code an ICHRFRTB macro instruction for the entry. Do not modify the IBM-supplied table, but, instead, create your own table called ICHRFR01.

ICHRFRTB concatenates the values specified for the REQSTOR, SUBSYS, and CLASS operands to form a 24-character string defining the entry. The macro matches these values against the string formed by the values specified on the RACROUTE macro instruction.

For information on coding the ICHRFRTB macro, see “ICHRFRTB Macro” in Chapter 10.

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

Selecting Options with SETROPTS

You can select the following options on the SETROPTS command:

- Universal access authority for terminals
- General resource protection
- Generic profile checking
- Global access checking
- Maximum password change interval
- Password syntax rules
- List-of-groups authority checking
- Refreshing of in-storage generic profile and global access checking lists
- Extended password processing
- Data set modeling options
- Bypassing automatic data set protection (ADSP)
- Bypassing RACINIT statistics
- Bypassing resource statistics
- Logging RACF command and RACDEF SVC activity
- Use of real data set names in messages and SMF records
- Bypassing logging of activity of users with the SPECIAL attribute
- Bypassing logging of RACF command violations
- RACF protection for data sets with single-level names
- JES2 or JES3 RACF support

- Activating tape data set protection
- Selecting a security retention period for tape data sets
- RACF-protecting all new data sets
- Erasing scratched DASD data sets
- Logging the activities of users with the OPERATIONS attribute
- Activating program control
- Setting the RVARY passwords
- Activating security classification of users and data

The security administrator is the primary user of the options on the SETROPTS command. For a description of these options, see the *Security Administrator's Guide*. For a complete description of the SETROPTS command, see the *Command Language Reference*.

Using Started Procedures

Started procedures have system-generated JOB statements that do not contain the USER, GROUP, or PASSWORD parameter. However, only users and groups can be specifically authorized to access RACF-protected resources. To give started procedures the same ability, you can use the RACF started procedures table (ICHRIN03) to associate the names of started procedures with RACF userids and group names. This option is part of a process that allows started procedures, such as JES, to have specific authorization to access RACF-protected resources, such as spool data sets. (If a started procedure uses a RACF-protected resource, you **must** define the started procedure to RACF in the started procedures table, ICHRIN03.)

As with any other userid and group name, the userid and group name that you assign to a started procedure must be defined to RACF using the ADDUSER and ADDGROUP commands. You may also need to authorize the users and/or groups to the required resources using the PERMIT command. See the *Command Language Reference* for descriptions of the commands.

The started procedure name is always available to the exit routines, whether or not the name is coded in the module. It is available in the parameter list for the RACINIT SVC and in the ACEE control block for the RACHECK and RACDEF SVCs.

If a started procedure is executed without associating its name with a RACF-defined userid and group name, RACF uses the default userid (*) and group name (*) for authorization checking. The procedure can access RACF-protected resources if the universal access authority for the resource is sufficient to allow the requested operation. No user verification (password checking) takes place for a started procedure's userid.

RACF allows the started procedures table to contain a generic entry, indicated by an asterisk (*) in the procedure name field. When searching the table for a procedure name match, if RACF finds a procedure name of "*" as the last entry in the table and the procedure name was not specifically matched by any other entry in the table, RACF uses the "*" entry as a match for the procedure name.

RACF also allows you to specify individual entries in the started procedures table as "privileged"; this means that all RACHECKs done for those entries are considered successful, without actually performing any checking. In these cases, RACF does not call any exit routines, does not generate any SMF records, and does not update any statistics. This bypassing also applies to the checking done for the CHKAUTH operand on the RACDEF macro instruction. All other RACF processing occurs as usual.

Note: If "ENTITY=(...,CSA)" is coded on the RACHECK macro instruction, RACF ignores the privileged attribute and performs normal RACHECK processing.

Note: A started procedure that has the privileged attribute bypasses any checking done by RACHECK, including category and security level checking. It can also access resources during failsoft processing without having RACF prompt the operator for permission.

Coding the Started Procedures Module

The RACF program product contains a module (ICHRIN03) that you may replace with your own table to associate the names of started procedures with userids and group names. There are no entries in the module when you receive it from IBM.

The table becomes part of the link pack area. Therefore, after replacing the module, you must re-IPL the system with the CLPA option for the new module to be in effect. (You could also load the module into the MLPA, so that the link pack area does not have to be recreated.)

The module (ICHRIN03) should consist of a table in the following format:

- **Number of entries** in the following array - a halfword of binary data, with the high-order bit turned on to indicate the new format used in RACF Version 1 Release 6 or later. (Use X'0000' or X'8000' if there are no entries.)

- **An array** - Each entry consists of 32 bytes of data. The first 24 bytes of character data show the started procedure name and its associated userid and group name. Format each entry as follows:

  - Started procedure name - eight bytes of character data. The name is required. The started procedure name must be left justified and padded on the right with blanks.

  - Userid - eight bytes of character data. A userid is required. The userid (or an equal sign for generic entries) must be left justified and padded on the right with blanks. (The maximum length of a userid is eight characters.)

  - Group name - eight bytes of character data. The group name is optional. If a group name (or an equal sign for generic entries) is used, it must be left justified and padded on the right with blanks. If a group name is not used, this field must contain blanks.

- Flags - one byte of binary data. Setting bit 0 on (X‘80’) indicates that this entry has the “privileged” attribute. The remaining seven bits must be zeroes. (See note.)
- Reserved - seven bytes of binary data. These seven bytes must be binary zeroes.

Note: RACF accepts started procedures tables (including the use of generic entries) in the format used in RACF releases prior to Release 6. However, you cannot use the old format if you want to specify the privileged attribute.

Generic Entries

The started procedures table can contain one generic entry, indicated by an asterisk (*) in the procedure name field. This entry must be the last entry in the table; otherwise, it is ignored. The corresponding userid in this entry can be a valid userid or an equal sign (=). The group name specified in the table entry can be either blanks, a valid group name, or an equal sign (=).

Note: You can use the equal sign only for a generic started procedures table entry; it is invalid for non-generic entries.

When searching the table for a procedure name match, if RACF finds a procedure name of “*” as the last entry in the table and the procedure name was not specifically matched by any other entry in the table, RACF uses the “*” entry as a match for the procedure name.

If a userid is specified for the “*” entry, RACF associates that userid with the started procedure name. If the userid field contains an “=”, RACF uses the procedure name that was matched with the generic entry (*) as the userid.

If the group name is blank, the started procedure will execute using the default group in the profile record for the specified userid (specified on the ADDUSER command). If the group name field contains an “=”, RACF uses the procedure name that was matched with the generic entry (*) as the group name.

If the generic entry has an “=” for the userid (or group name), the procedure name that matches the equal sign must be defined to RACF as a userid (or group name); otherwise the procedure runs as an undefined RACF user (userid = *).

The userid and the group name cannot both contain values of “=” in the “*” procedure name entry of the table because it is not possible to have a RACF user and group with the same name. During RACF initialization, ICHSEC00 inspects the table entries for a possible generic entry. If ICHSEC00 finds a generic entry, and it is not the last entry, or if it contains an “=” in both the userid and group name fields, the system issues message ICH522I. This condition does not prevent RACF from being initialized. During execution, RACF ignores all the invalid entries, and all procedures that do not have an exact match in the table run as undefined users.

If you do not specify an "*" in the table, RACF uses the RACF default userid (*) and group name (*) for authorization checking.

Note: You should specify a password on the ADDUSER command for a started procedure. If you do not specify a password, RACF uses the userid default group as the password. Any user who knows the started procedure's default group can use the userid and default password to access the system.

The started procedures table (ICHRIN03) can include an entry indicated by an "*" in the procedure name field as the last entry in the table. The following examples show the possible formats of the "*" procedure name entry. Note that none of these examples has the "privileged" flag bit on.

Example 1

COUNT	PROC.	USERID	GROUP	FLAGS	RESERVED
X'8002'	PROC1	TSO1	SYS1	00000000	7 bytes of X'00'
	*	TSO2	=		

If RACF searched the started procedures table in Example 1 for the procedures name PROC2, it would not find a specific match. RACF would consider the "*" entry in the table as a match for procedure PROC2. The userid associated with PROC2 is TSO2, and the group name is PROC2.

RACF associates PROC1 with userid TSO1 and group SYS1.

Example 2

COUNT	PROC.	USERID	GROUP	FLAGS	RESERVED
X'8002'	PROC1	TSO1	SYS1	00000000	7 bytes of X'00'
	*	=			

If RACF searched the started procedures table in Example 2 for the procedures name PROC2, it would not find a specific match. RACF would consider the "*" entry in the table as a match for procedure PROC2. The userid associated with PROC2 is PROC2, and the group name is the default group defined for RACF user PROC2.

Example 3

COUNT	PROC.	USERID	GROUP	FLAGS	RESERVED
X'8001'	PROC1	TSO1	SYS1	00000000	7 bytes of X'00'

If RACF searched the started procedures table in Example 3 for the procedures name PROC2, it would not find a specific match. RACF would associate with PROC2 the default userid (*) and the default group name (*).

RACF associates PROC1 with userid TSO1 and group SYS1.

Example 4

COUNT	PROC.	USERID	GROUP	FLAGS	RESERVED
X'8002'	*	TSO2	SYS2	00000000	7 bytes of X'00'
	PROC1	TSO1	SYS1		

If RACF searched the started procedures table in Example 4 for the procedures name PROC2, it would not find a specific match. Because the last entry in the table does not contain an "*" procedure name, RACF issues an error message during RACF initialization. RACF ignores the generic entry during RACINIT processing.

RACF associates PROC1 with userid TSO1 and group SYS1.

Using the Data Encryption Standard (DES) Algorithm

If activated, RACF uses the Data Encryption Standard (DES) algorithm to encrypt passwords and OIDCARD data before they are stored in the RACF data set. This algorithm is more secure than the "masked" form used in releases of RACF prior to Release 6.

RACF performs two different encryption functions:

- Password/OIDCARD data comparison

Encryption means that, given data and an encryption key, RACF produces the equivalent data in encrypted form. Normally, encryption and decryption are the inverse of each other; data that has been encrypted can also be decrypted provided that you know the encryption key. However, RACF creates a one-way encryption; that is, you can decode the encrypted data only if you already know the data.

Comparison means that, given a clear text password (or OIDCARD data) entered by a user and a password stored in the RACF data set (in encrypted form), RACF determines whether the two are equal or not. RACF handles this comparison in the following way:

1. Encrypts the data using the DES algorithm and compares it to the stored version. If they are equal, RACF returns to the caller with an "equal" indication.

2. Encrypts the data using the current masking algorithm and compares it to the stored version. RACF returns to the caller with an "equal" or "not equal" indication.

This procedure allows the use of existing masked passwords and OIDCARD data until they are all replaced by the DES forms. This replacement eventually occurs, because whenever a new password is given, RACF encrypts the new password and stores it in the RACF data set in the encrypted form.

You can use the exit routine ICHDEX01 to replace the DES algorithm with any algorithm you want (including the masking algorithm used in prior RACF releases). For more information on this exit, see Chapter 7.

For compatibility with prior RACF releases, RACF has a dummy ICHDEX01 exit routine. This dummy exit routine always sets a return code of 4, telling RACF to use only the masking algorithm. To activate the DES algorithm, you must delete the dummy exit from LPALIB and do an IPL. You should perform this procedure on all systems sharing the RACF data set after they have been converted to a version of RACF that supports the DES algorithm.

For information on activating the DES algorithm, see "Activating the Data Encryption Standard Algorithm" in Chapter 1.

There is a slight possibility that the DES-encrypted form of a password is identical to the masked form of another password. Thus, because a masked password can be decrypted, a devious user who happens to get access to the DES-encrypted form of a password may be able to find another password that will be accepted during logon because of the two comparisons. Once all masked data has been converted to the DES-encrypted form, you may want to install an ICHDEX01 exit that always sets a return code of 8, instructing RACF to ignore the masking algorithm and use only the DES algorithm.

Notes:

1. *When the user is on TSO, the terminal status block (TSB) control block still stores the clear text form of the password in read-protected storage for the duration of the terminal session.*

2. *Currently, the only IBM-defined fields in the RACF data set that RACF always encrypts are the PASSWORD, OLDPWD, and MAGSTRIP fields in the user profiles. But, in addition,* **any** *profile field that has bit 5 (X'04') on in the template definition is also handled in the same way. However, when encrypting such fields, RACF uses the first eight characters of the profile name rather than the userid. Therefore, it is not recommended that fields in the data set profiles be encrypted because these profiles can be renamed and thus make any comparison of the encrypted fields impossible.*

   In addition to the template-controlled option, you can instruct the ICHEACTN and ICHETEST macros (by the ENCRYPT=YES|NO|TEMPLATE keyword) to always, or never, encrypt a data field.

3. *If you choose to delete ICHDEX01 in order to activate DES, be sure to delete it again if you re-install RACF.*

Changing the ICHAUTAB Module

The RACF authorized-caller table contains the names of programs that your installation authorizes to issue the RACINIT SVC without the NEWPASS keyword or the RACLIST SVC. This table resides in the link pack area (LPA) in ICHAUTAB. ICHAUTAB is an installation-replaceable module.

For each authorized caller (program), the RACF authorized-caller table contains a 12-byte entry in the following format:

Length	Description
8	Caller name, left-justified and padded with blanks. (The last entry in the table must contain a blank caller name.)
4	Authorization code. X'40000000' - the caller is authorized to issue the RACLIST SVC. X'80000000' - the caller is authorized to issue the RACINIT SVC without the NEWPASS keyword.

The RACF authorized-caller table resides in the link pack area (LPA) in ICHAUTAB. To add an entry to the RACF authorized-caller table, you can do one of the following:

- Use the SPZAP service aid to add the entry to the IBM-supplied ICHAUTAB module.

  Note: ICHAUTAB can handle up to six table entries. If your installation requires more than six, you must reassemble the ICHAUTAB module.

- Reassemble the ICHAUTAB module with the new entry and linkedit it again into the LPA.

The following example shows how an installation can use the RACF authorized-caller table:

Example: An installation wants its VTAM application, Network Communications Control Facility (NCCF) (program number 5735-XX6) to be authorized to issue the RACINIT SVC without the NEWPASS keyword so that NCCF can perform additional security checks on the user's password. To do this, the installation codes the following in the RACF authorized-caller table:

Coded	Description
CL8'DSIOST'	NCCF program name
XL4'80000000'	NCCF is authorized to issue calls to the RACINIT SVC without the NEWPASS keyword.

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 userid, number and type of successful accesses, and number and type of unauthorized access attempts.
- Produce reports that describe user and group activity.
- Produce reports that summarize system use and resource use.

For more information on the RACF report writer and the RACF report writer command (RACFRW), see the *Auditor's Guide*.

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

The format of ICHRSMFI is:

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 non-blank characters in the SORTTBL parameter field.)	binary (0 or 256). Zero means ignore this table	0

Name	Offsets DEC(HEX)	Length	Description	Format	Default
SORTDYN	279(117)	32	SORT/MERGE dynamic allocation of intermediate work space parameter	EBCDIC and left-justified	DYNALLOC= 3350
SORTDYNS	311(137)	2	SORT/MERGE dynamic allocation parameter size. (This parameter equals the actual number of of non-blank characters in the SORTDYN parameter field.)	binary. Zero means no dynamic allocation by SORT/MERGE	13
SORTEQU	313(139)	8	SORT/MERGE preservation of input order for records with equal sort fields	EBCDIC (EQUALS or NOEQUALS)	NOEQUALS
SORTEQUS	321(141)	2	SORT/MERGE SORTEQU field size. (This parameter equals the actual number of non-blank characters in the SORTEQU parameter field.)	binary (6 for EQUALS and 8 for NOEQUALS)	8
SORTDSN	323(143)	44	SORT/MERGE SORTLIB data set name	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 and at least 1024	1024
WRKBLKSI	380(17C)	2	SORTIN block size	binary	3256
WRKUNIT	382(17E)	8	SORTIN unit	EBCDIC, left-justified, and padded with blanks. All blanks mean information obtained from Protected Step Control Block	SYSDA
WRKSER	390(186)	6	SORTIN volume serial	EBCDIC, left-justified, and padded with blanks. All blanks mean no specific volume serial	all blanks
INBUFFSI	396(18C)	4	Size of internal buffer for rebuilding SMF records	binary	2048
INITREC	400(190)	1	SMF record type used for job initiation / TSO logon recording	binary	20

For additional information on the SORT/MERGE parameters, see the *OS/VS Sort/Merge Programmer's Guide*, SC33-4035.

You should review the defaults in ICHRSMFI to ensure that they apply to your current operating environment. For example, in ICHRSMFI, the default for SORTDYN is DYNALLOC=3350, and the default for SORTDYNS is 13. If your installation uses 3380 devices, you must change the SORTDYN default to DYNALLOC=3380. Otherwise, running the RACF report writer at your installation would cause the SORT/MERGE module to terminate with a SORT/MERGE return code of 16 and a dynamic allocation return code of 218.

To change the ICHRSMFI default values, use the service aid SPZAP to add the new values to the ICHRSMFI module.

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

Chapter 6. RACF Utilities

RACF provides the following utilities for maintaining, modifying, and monitoring the RACF data set(s):

- RACF data set initialization utility program (ICHMIN00) - formats a non-VSAM DASD data set for use as a RACF data set. You use this utility during the initial installation of RACF and when converting a Release 6 or earlier RACF data set to a Release 7 RACF data set.

- RACF cross-reference utility program (ICHUT100) - lists all the occurrences of a userid or group name that exist in the RACF data set. The security administrator is the primary user of this utility.

- RACF data set verification utility program (ICHUT200) - identifies inconsistencies in the internal organization of a RACF data set and provides information about the size and organization of a RACF data set. You can use this utility to create a backup copy of a RACF data set.

- Block update utility command (ICHUT300) - modifies the records in a RACF data set. You execute ICHUT300, as a RACF command (BLKUPD), to correct any inconsistencies that the RACF data set verification utility finds in the RACF data set.

- Split/merge/extend utility program (ICHUT400) - splits a single existing RACF data set into multiple data sets, combines or redistributes multiple data sets, or copies an existing RACF data set to a larger data set. This utility allows you to distribute your RACF data, according to your selection criteria, across up to 255 output data sets.

RACF Data Set Initialization Utility Program (ICHMIN00)

The RACF data set initialization program (ICHMIN00) formats a non-VSAM DASD data set so that it can be used as the RACF data set.

If you have a prior release of RACF installed, you must always use the latest release of ICHMIN00 to which you are migrating in order to update the existing RACF data set. If you wish to copy the program, make the copy first and then use ICHMIN00, which updates the templates to the new RACF release level.

ICHMIN00 requires one of the following parameters on the JCL EXEC statement:

- PARM=NEW - specified when formatting a new RACF data set
- PARM=UPDATE - specified when formatting a RACF data set from a previous release. ICHMIN00 formats the data set with the old profiles left intact.

You must run ICHMIN00 during the initial installation of RACF to format the RACF data set. After RACF is installed, you can run ICHMIN00 to format an alternate RACF data set.

The ICHMIN00 program divides a RACF data set into 1K byte records. When you create a new RACF data set, the following records are initialized:

Record	Description
ICB block	The header block (index control block).
Templates	Ten template records - the user, group, connect, data set, and general template definitions, plus five reserved records.
BAM blocks	BAM (block availability mask) blocks are initialized with the space configuration for the data set.
Index block	One index block is initialized with an entry of X'FF' to indicate there are no entity records.

Note: No profiles are initialized.

When you update an existing RACF data set, the ICB block is updated to reflect the new fields and the templates are replaced by the new templates. The BAM blocks, index blocks, and profiles are not altered.

Input

ICHMIN00 requires the following DD statements with ddnames of:

Ddname	Description
SYSTEMP	Defines the ddname for the template definition data set (ICHTEMP0 on SYS1.MACLIB) that ICHMIN00 uses to initialize the template definition records in a RACF data set. ICHMIN00 checks the validity of the template definition data set when it initializes the template definition records.
SYSPRINT	Defines the output data set. The minimum block size is 129 (enforced by a DCB exit).
SYSRACF	Defines a contiguous, unmoveable non-VSAM data set to be formatted. The logical record size and block size are 1024 (enforced by a DCB exit). If you are updating an existing RACF data set, ICHMIN00 updates the RACF data set in place.

For an example of the JCL required to allocate space, catalog the data set, and run the ICHMIN00 program, see the RACF program directory.

Output

The input images from the template definition data set (SYSTEMP) are written to the output data set (SYSPRINT) along with messages indicating errors or success.

The ICHMIN00 program sets the following return codes:

Hex (Decimal)	Meaning
0 (0)	Successful completion.
4 (4)	Warning - the RACF data set is usable but the contents of the template definition data set should be verified.
C (12)	The program encountered a terminating error. The RACF data set was not formatted.
10 (16)	The output data set defined by DD SYSPRINT could not be opened. The RACF data set was not formatted.

RACF Cross-Reference Utility Program (ICHUT100)

ICHUT100 is a RACF utility program that lists all occurrences of a userid or group name that are in a RACF data set. It uses the RACF manager to access the RACF data set and locate possible occurrences of a userid or group name.

To invoke ICHUT100, you must be a RACF-defined user and either have the SPECIAL, group-SPECIAL, AUDITOR, or group-AUDITOR attributes, or be requesting a list of occurrences for only your userid.

ICHUT100 produces a cross-reference report that describes the occurrences of each userid or group name specified. The letter G in parentheses follows each generic profile name.

The following information is provided about a group:

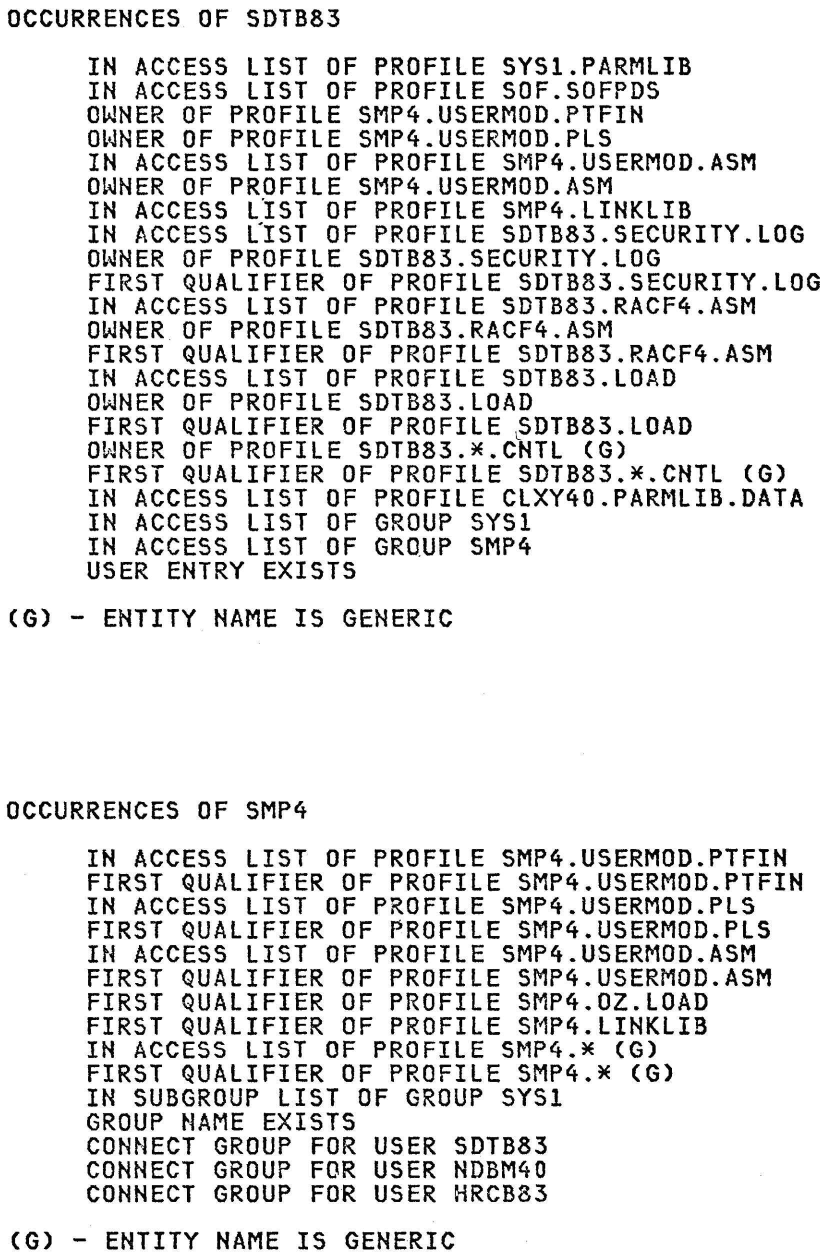
- Whether the group name exists in the RACF data set.
- The group is a subgroup of group xx.
- The group is a superior group of group xx.
- The group is the default group for user xx.
- The group is a connect group for user xx.
- The group was the connect group when the user created data set profile xx.
- The group name is the high-level qualifier of data set profile xx.
- The group has access to data set profile xx.
- The group has 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 following information is provided about a user:

- Whether the userid exists in the RACF data set.
- 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 access to data set profile xx.
- The user has access to general resource xx.
- The userid 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.

See Figure 6-1 for a sample output of the printed report that ICHUT100 produces.



```
OCCURRENCES OF SDTB83

      IN ACCESS LIST OF PROFILE SYS1.PARMLIB
      IN ACCESS LIST OF PROFILE SOF.SOFPDS
      OWNER OF PROFILE SMP4.USERMOD.PTFIN
      OWNER OF PROFILE SMP4.USERMOD.PLS
      IN ACCESS LIST OF PROFILE SMP4.USERMOD.ASM
      OWNER OF PROFILE SMP4.USERMOD.ASM
      IN ACCESS LIST OF PROFILE SMP4.LINKLIB
      IN ACCESS LIST OF PROFILE SDTB83.SECURITY.LOG
      OWNER OF PROFILE SDTB83.SECURITY.LOG
      FIRST QUALIFIER OF PROFILE SDTB83.SECURITY.LOG
      IN ACCESS LIST OF PROFILE SDTB83.RACF4.ASM
      OWNER OF PROFILE SDTB83.RACF4.ASM
      FIRST QUALIFIER OF PROFILE SDTB83.RACF4.ASM
      IN ACCESS LIST OF PROFILE SDTB83.LOAD
      OWNER OF PROFILE SDTB83.LOAD
      FIRST QUALIFIER OF PROFILE SDTB83.LOAD
      OWNER OF PROFILE SDTB83.*.CNTL (G)
      FIRST QUALIFIER OF PROFILE SDTB83.*.CNTL (G)
      IN ACCESS LIST OF PROFILE CLXY40.PARMLIB.DATA
      IN ACCESS LIST OF GROUP SYS1
      IN ACCESS LIST OF GROUP SMP4
      USER ENTRY EXISTS

(G) - ENTITY NAME IS GENERIC




OCCURRENCES OF SMP4

      IN ACCESS LIST OF PROFILE SMP4.USERMOD.PTFIN
      FIRST QUALIFIER OF PROFILE SMP4.USERMOD.PTFIN
      IN ACCESS LIST OF PROFILE SMP4.USERMOD.PLS
      FIRST QUALIFIER OF PROFILE SMP4.USERMOD.PLS
      IN ACCESS LIST OF PROFILE SMP4.USERMOD.ASM
      FIRST QUALIFIER OF PROFILE SMP4.USERMOD.ASM
      FIRST QUALIFIER OF PROFILE SMP4.OZ.LOAD
      FIRST QUALIFIER OF PROFILE SMP4.LINKLIB
      IN ACCESS LIST OF PROFILE SMP4.* (G)
      FIRST QUALIFIER OF PROFILE SMP4.* (G)
      IN SUBGROUP LIST OF GROUP SYS1
      GROUP NAME EXISTS
      CONNECT GROUP FOR USER SDTB83
      CONNECT GROUP FOR USER NDBM40
      CONNECT GROUP FOR USER HRCB83

(G) - ENTITY NAME IS GENERIC
```

Figure 6-1. Sample Output from ICHUT100

Input and Output

ICHUT100 uses as input a control data set. The control data set contains the utility control statements that indicate which names to cross-reference.

ICHUT100 produces the following output:

- A message data set - containing the results of the ICHUT100 operations. The message data set includes the printed report and any error messages.
- A work data set - containing the internal records describing the occurrences of each input name. This work file provides the data for the listed report and may be kept at the end of the job for other applications.

Control

ICHUT100 is controlled by job control statements and utility control statements. The job control statements are necessary to execute or invoke the program and to define the data sets used and produced by the program. The utility control statements specify the names to be cross-referenced.

Job Control Statements

The following job control statements are necessary for using ICHUT100.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM = ICHUT100) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to an output device, a tape volume, or a direct access device.
SYSUT1 DD	Defines a work data set on a direct access device.
SYSIN DD	Defines the control data set. The control data set is normally found in the input stream; however, it can be a member of a procedure library or a sequential data set existing elsewhere.

Note: If the utility is executed under TSO, you can allocate both the SYSIN and SYSPRINT data sets to the terminal.

Utility Control Statements

ICHUT100 control statements have the following format:

```
name  [name]
```

where:

name is a group name or userid that is one to eight characters long and begins in any column.

Names are separated by either 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
```

where:

/END terminates the utility program. The /END statement is not required. If it is coded, this statement must begin in column 1. An end-of-file on the SYSIN data set also terminates the program.

The Work Data Set

Records in the work data set are 56 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 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 access list of resource name.
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.

Bytes 4-55:	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.

Exit Routine

RACF provides a preprocessing exit for an installation-written routine when the ICHUT100 utility is invoked. Before calling the ICHUT100 exit, RACF processes the data set naming convention table. The exit routine is entered after syntax checking and after the data set profile or generic profile is retrieved, but before the profile is associated with a user or group. You can specify the user or group to be associated with this profile. For more information, see "Command Preprocessing Routine ICHCNX00" in Chapter 7.

ICHUT100 Example

In this example, ICHUT100 locates all occurrences of the group name RACG0001 and the userid RACU002 in the RACF data set and prints these occurrences on the system output device.

```
//XREF       JOB    USER=RAC01,PASSWORD=xxx
//STEP       EXEC   PGM=ICHUT100
//SYSUT1     DD     UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSPRINT   DD     SYSOUT=A
//SYSIN      DD     *
 SDTB83      SAMPL4
/END
```

RACF Data Set Verification Utility Program (ICHUT200)

ICHUT200 is a RACF utility program that identifies inconsistencies in the internal organization of a RACF data set. It performs the following functions:

- Scans the index blocks and prints information about problems with the index block chains
- Compares the segments of the data set that are actually in use to the segments allocated according to the BAM blocks, and prints information about inconsistencies
- Creates a backup copy of a RACF data set, if requested

If you specify a work data set in your JCL, ICHUT200 copies the RACF data set into this work data set. ICHUT200 reserves the RACF data set during the copying by using a RESERVE macro instruction with a request for shared control. After the copying is complete, the RACF data set is released.

If you do not specify a work data set, the RACF data set is not released after the copying is complete. It remains reserved for the duration of the processing.

If a DD statement for the RACF data set is not included in your JCL, the utility assumes that a copy already exists in the work data set.

If a RACF data set is RACF-protected, you must have at least READ authority to access the data set. ICHUT200 runs as an APF-authorized program.

When running the ICHUT200 utility under a TMP (terminal monitor program) that allows multitasking, you cannot have any other active task in your session. Allow ICHUT200 to complete before executing any other TSO command.

Scanning the Index Blocks

When an index block scan is requested, ICHUT200 performs the following verifications:

- The pointer to every index block must be a multiple of 1024.
- Every index block must begin with the value X‘8A’.
- Every index entry name must have a valid length.
- Every pointer entry in the index block must be preceded by the value X‘6x’ (x may be any value).
- Only level one blocks can appear in the sequence set.
- Offsets to the last index entry in each block must be correct.
- Offsets to free space in each index block must be correct.

Unformatted Printout

If an index block does not meet all the requirements during the verification process, ICHUT200 prints a dump of the block in hex. An error message precedes the dump.

Some of these messages are also printed in a message data set. For an explanation of these messages, see “RACF Data Set Verification Utility Program (ICHUT200) Messages” later in this chapter.

ICHUT200 provides the following information for each block that is not dumped:

- Title lines identifying the level and relative byte address (RBA) of the index block
- Validity check messages pertaining to the block
- The total number of entry names in the block
- The number of unused bytes in the block
- The average name length in the block
- The level of the block as defined in the header
- The offset to the last entry name in the block
- The offset to free space as defined in the header of the block

Following this information are summary statistics about the index. These statistics might not be representative of the entire data set because they represent only the processed blocks that were not dumped due to errors. The following summary statistics are:

- The total number of index entry names in a RACF data set. A name is counted each time it appears in the index.
- The average number of names per index block.
- The average name length in the entire index.
- The average number of unused bytes per 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, ICHUT200 also provides, in addition to the output previously described, a formatted printout of each index block that is not dumped because of an error. The formatted block immediately follows any validity check messages for that block. The following information is provided for each index entry within the block:

- The offset of the entry within the block
- The front-end compression count
- The entry name (generic entry names are followed by a G in parentheses)
- The last qualifier encode byte
- 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

Notes:

1. *For connect entries (such as IBMUSER/SYS1), the slash (/) represents X'00'.*
2. *See Figure 6-2 for sample output that ICHUT200 produces when you request formatted index blocks.*

```
                                         ****  INDEX BLOCK VERIFICATION  ****

                                     ****  SCAN OF INDEX BLOCKS AT LEVEL 01  ****
BLOCK WITH RBA OF 000000197000

OFFSET COMPRESSION                 ENTRY NAME                   ENCODE     RBA           BAM
          COUNT                                                  BYTE               BLOCK BYTE BIT
  00A    00    AWB632                                                  00000031E700   01   264  7
  01A    06    AWB632/M01632                                           000000073500   00   0FA  5
  02B    02    AWM095                                                  0000002A0600   01   168  6
  039    02    AWM095/M01PGM                                           0000000EC400   00   1EC  4
  04E    02    AWM095/M01095                                           000000101000   00   216  0
  063    00    A05                                                     000000324700   01   270  7
  070    00    A05.CMDLIB                                              00000000C500   00   02C  5
  084    00    A05.PGMLIB                                              000000325E00   01   273  6
  098    00    A05.PROCLIB                                             000000006400   00   020  4
  0AD    00    A05C                                                    00000020F500   01   046  5
  0BB    01    A05801                                                  00000001EF00   00   051  7
  0CA    01    A05952                                                  0000001F8900   01   019  1
  0D9    01    A05980                                                  00000001F100   00   052  1
  0E8    00    BACKER                                                  00000003AC00   00   089  4
  0F8    00    BACKER/M0153K                                           000000032F00   00   079  7
  10F    00    BAC1R2                                                  00000033F100   01   2A6  1
  11F    00    BAC1R2/RACFTEMP                                         000000344D00   01   2B1  5
  138    00    BAL1AS                                                  00000044D000   02   0D6  0
  148    00    BAL1AS/RACFTEMP                                         00000044D100   02   0D6  1
  161    00    BAL83K                                                  000000016800   00   041  0
  171    00    BAL83K/RACFTEMP                                         00000000BF00   00   02B  7
  18A    00    BARNWEL                                                 000000317300   01   256  3
  19B    00    BARNWEL/RACFTEST                                        000000029400   00   066  4
  1B5    00    BASE2                                                   000000059A00   00   0C7  2
  1C4    00    BASE2/SYS1                                              00000005AB00   00   0C9  3
  1D8    00    BASE2.* (G)                                             00000026E500   01   104  5
  1EA    00    BCBB86                                                  0000002F8E00   01   219  6
  1FA    00    BCBB86/B86IAR                                           00000000DCB00  00   1CD  3
  211    00    BCBB86/M02B86JM                                         0000001E3200   00   3DA  2
  22A    00    BCBB86/PARM                                             0000001DBE00   00   3CB  6
  23F    00    BCBB86/PL                                               0000002F8500   01   218  5
  252    00    BCBB86/RELEASE                                          0000001A6900   00   361  1
  26A    00    BCBB86/R510                                             000000108700   00   224  7
  27F          SEQUENCE SET POINTER                                    000000130400

TOTAL NAMES IN THIS BLOCK-033. UNUSED BYTES-0376. AVERAGE NAME LENGTH-09.
LEVEL NUMBER-01. DISPLACEMENT TO LAST KEY-027F. DISPLACEMENT TO FREE SPACE-0288.
(G) - ENTITY NAME IS GENERIC
```

Figure 6-2 (Part 1 of 2). Sample Output of Formatted Index Produced by ICHUT200

```
                                                 ****  SEQUENCE SET RBAS  ****
RBA   000000004400
RBA   000000167C00
RBA   0000002A8800
RBA   0000001E5C00
RBA   00000036A000
RBA   0000001D1000
RBA   00000034B000
RBA   0000003B0000
RBA   000000341400
RBA   000000306400
RBA   00000036E000
RBA   000000309000
RBA   00000007A400
RBA   000000197000
RBA   000000130400
RBA   000000446800
RBA   000000046000
RBA   00000038D000
RBA   000000372000
RBA   000000340800
RBA   000000387000
RBA   000000342800
RBA   0000003A6400
RBA   0000003A5000
RBA   0000003A3C00
RBA   000000166C00
RBA   000000346C00
RBA   000000337000


                                                   ****  INDEX FUNCTION STATISTICS  ****

TOTAL NUMBER OF NAMES IN RACF DATA SET 00020431
TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000797
AVERAGE NUMBER OF NAMES PER INDEX BLOCK 025
AVERAGE NAME LENGTH 09
AVERAGE NUMBER OF UNUSED BYTES PER INDEX BLOCK 0497
TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000773
```

Figure 6-2 (Part 2 of 2). Sample Output of Formatted Index Produced by ICHUT200

BAM/Allocation Comparison

When a BAM/allocation comparison is requested, ICHUT200 performs the following verifications:

- Every index entry name must have a valid length.
- 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 index control block (ICB).
- The RBA of each template defined in the ICB must have these characteristics:
  - It is a multiple of 1024.
  - The first two bytes are zero.
  - The last four bytes are nonzero.
- The RBA of each BAM block must be a multiple of 1024 and its first two bytes must be zero.
- The count of BAM blocks in the ICB must be greater than zero.
- The number of blocks defined by a BAM block must be from 1 to 2008.

When a block does not meet all of these requirements, ICHUT200 prints a dump of the block in hex. An error message precedes the dump.

Some of these messages are also printed in a message data set. For an explanation of these messages, see "RACF Data Set Verification Utility Program (ICHUT200) Messages" later in this chapter.

ICHUT200 produces an encoded map of each BAM block. Each map is identified by a block number and its relative byte address (RBA), and contains byte offsets to the coded masks within the block. The codes indicate the type of block and the types of consistencies or inconsistencies that exist between the actual allocation of data set segments and the status of the segments as defined by the masks in the BAM blocks. 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 *.
B	Refers to a BAM block. This symbol implies *.
T	Refers to a template block. This symbol implies *.
F	Refers to the first block (ICB). This symbol implies *.

–	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 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.

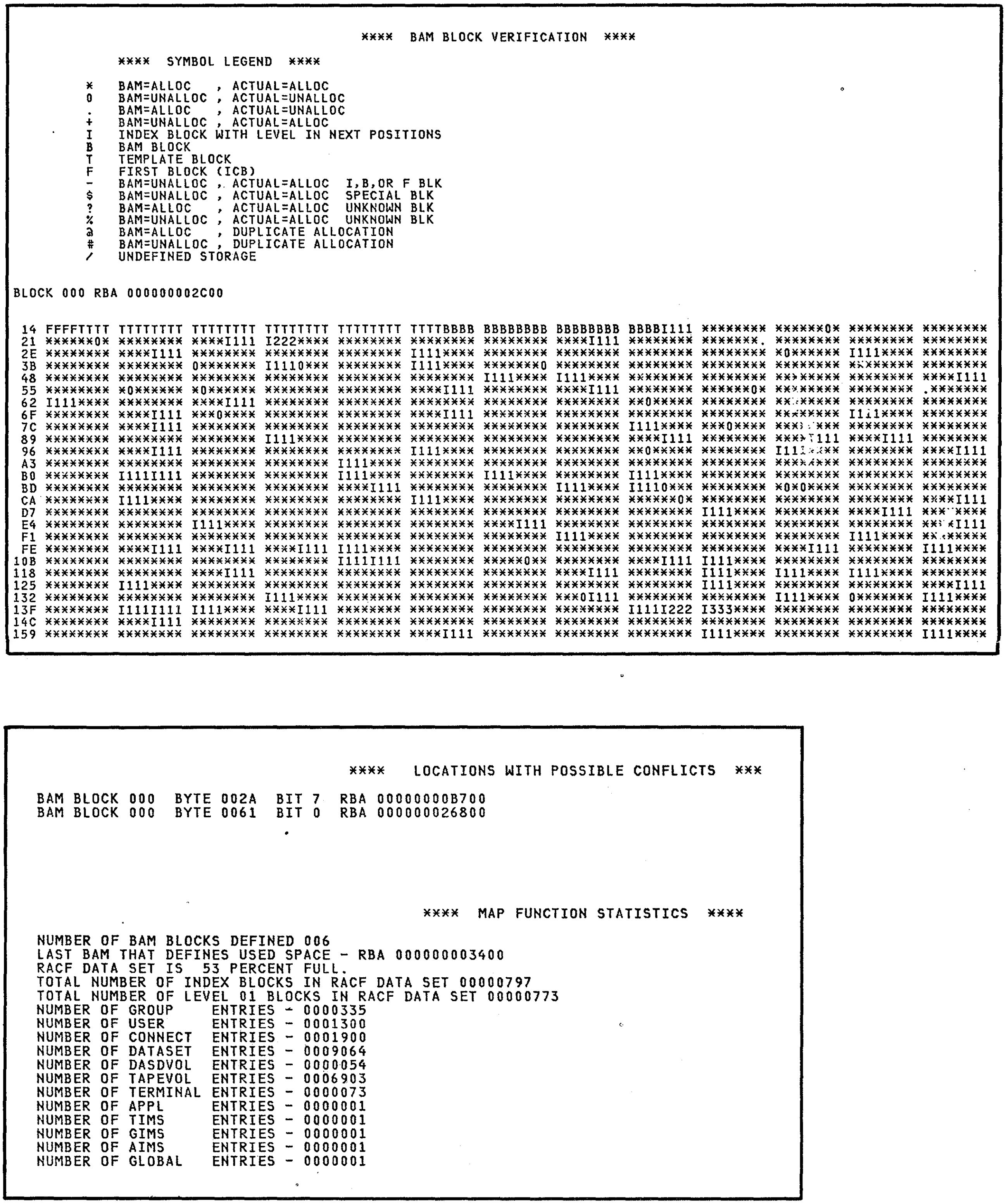
Following the encoded blocks, ICHUT200 prints a table of conflict messages that lists the first 200 locations of possible conflicts in the BAM blocks. These messages locate the inconsistencies by referencing the corresponding block, byte, and bit of the encoded mappings. Each word of the encoded map represents one byte of the BAM block. The relative byte address (RBA) of the storage represented by the bit is also included.

ICHUT200 also provides the following summary statistics concerning the RACF data set:

- The number of BAM blocks defined in the ICB
- The RBA of the last BAM block that defines used space
- The total number of index blocks in the data set
- The total number of level one index blocks
- The number of profiles of each type in the data set
- The percentage of space used on a RACF data set

ICHUT200 produces an encoded map for every BAM block, whether inconsistencies are found or not. As an option, you can request that the encoded maps for an entire RACF data set be printed. If inconsistencies are found, a table of conflict messages follows.

See Figure 6-3 for a sample printout of the encoded map that ICHUT200 produces.



```
                                        ****  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
        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 000000002C00

 14 FFFFTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTTTTT TTTTBBBB BBBBBBBB BBBBBBBB BBBBI111 ******** ******0* ******** ********
 21 ******0* ******** ****I111 I222**** ******** ******** ******** ****I111 ******** *******. ******** ******** ********
 2E ******** ****I111 ******** ******** ******** I111**** ******** ******** ******** ******** *0****** I111**** ********
 3B ******** ******** 0******* I1110*** ******** I111**** *******0 ******** ******** ******** ******** ******** ********
 48 ******** ******** ******** ******** ******** ******** I111**** I111**** ******** ******** ******** ******** ****I111
 55 ******** *0****** *0****** ******** ******** ****I111 ******** ****I111 ******** ******0* ******** ******** .*******
 62 I111**** ******** ****I111 ******** ******** ******** ******** ******** **0***** ******** ******** ******** ********
 6F ******** ****I111 ***0**** ******** ******** ****I111 ******** ******** ******** ******** ******** I111**** ********
 7C ******** ****I111 ******** ******** ******** ******** ******** ******** I111**** ***0**** ******** ******** ********
 89 ******** ******** ******** I111**** ******** ******** ******** ******** ****I111 ******** ****I111 ****I111 ********
 96 ******** ****I111 ******** ******** ******** I111**** ******** ******** **0***** ******** I111**** ******** ****I111
 A3 ******** ******** ******** ******** I111**** ******** ******** ******** ******** ******** ******** ******** ********
 B0 ******** I111I111 ******** ******** I111**** ******** I111**** ******** I111**** ******** ******** ******** ********
 BD ******** ******** ******** ******** ****I111 ******** ******** I111**** I1110*** ******** *0*0**** ******** ********
 CA ******** I111**** ******** ******** ******** I111**** ******** ******** ******0* ******** ******** ******** ****I111
 D7 ******** ******** ******** ******** ******** ******** ******** ******** ******** I111**** ******** ****I111 ********
 E4 ******** ******** I111**** ******** ******** ******** ****I111 ******** ******** ******** ******** ******** ****I111
 F1 ******** ******** ******** ******** ******** ******** ******** I111**** ******** ******** ******** I111**** ********
 FE ******** ****I111 ****I111 ****I111 I111**** ******** ******** ******** ******** ******** ****I111 ******** I111****
10B ******** ******** ******** ******** I111I111 ******** *****0** ******** ****I111 I111**** ******** ******** ********
118 ******** ******** ****I111 ******** ******** ******** ******** ****I111 ******** I111**** I111**** I111**** ********
125 ******** I111**** ******** ******** ******** ******** ******** ******** ******** I111**** ******** ******** ****I111
132 ******** ******** ******** I111**** ******** ******** ******** ***0I111 ******** ******** I111**** 0******* I111****
13F ******** I111I111 I111**** ****I111 ******** ******** ******** ******** I111I222 I333**** ******** ******** ********
14C ******** ****I111 ******** ******** ******** ******** ******** ******** ******** ******** ******** ******** ********
159 ******** ******** ******** ******** ******** ****I111 ******** ******** ******** I111**** ******** ******** I111****
```

```
                                   ****   LOCATIONS WITH POSSIBLE CONFLICTS  ***

  BAM BLOCK 000   BYTE 002A   BIT 7   RBA 00000000B700
  BAM BLOCK 000   BYTE 0061   BIT 0   RBA 000000026800



                                          ****  MAP FUNCTION STATISTICS   ****

  NUMBER OF BAM BLOCKS DEFINED 006
  LAST BAM THAT DEFINES USED SPACE - RBA 000000003400
  RACF DATA SET IS  53 PERCENT FULL.
  TOTAL NUMBER OF INDEX BLOCKS IN RACF DATA SET 00000797
  TOTAL NUMBER OF LEVEL 01 BLOCKS IN RACF DATA SET 00000773
  NUMBER OF GROUP    ENTRIES - 0000335
  NUMBER OF USER     ENTRIES - 0001300
  NUMBER OF CONNECT  ENTRIES - 0001900
  NUMBER OF DATASET  ENTRIES - 0009064
  NUMBER OF DASDVOL  ENTRIES - 0000054
  NUMBER OF TAPEVOL  ENTRIES - 0006903
  NUMBER OF TERMINAL ENTRIES - 0000073
  NUMBER OF APPL     ENTRIES - 0000001
  NUMBER OF TIMS     ENTRIES - 0000001
  NUMBER OF GIMS     ENTRIES - 0000001
  NUMBER OF AIMS     ENTRIES - 0000001
  NUMBER OF GLOBAL   ENTRIES - 0000001
```

Figure 6-3. Sample Output of Encoded BAM Map Produced by ICHUT200

Input and Output

ICHUT200 uses the following input:

- A control data set, which contains the utility control statements that indicate the functions to be performed.
- A RACF data set. (This data set is not required if the work data set already contains a copy.)
- A work data set into which a RACF data set is copied. (This data set is not required if a RACF data set is to be used throughout processing.) Note that this work copy can be used as a backup RACF data set.

ICHUT200 produces the following output:

- A message data set - containing diagnostic error messages.
- An output data set - for printing statistical data and the results of the ICHUT200 operations.

Control

ICHUT200 is controlled by job control statements and utility control statements. The job control statements are necessary to execute or invoke the program and define the data sets used and produced by the program. The utility control statements control the functions of the program.

Job Control Statements

The following job control statements are necessary for using ICHUT200:

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM = ICHUT200) or, if the job control statements reside in a procedure library, the procedure name.
SYSRACF DD	Defines a RACF data set on a direct access device. ICHUT200 requires this data set unless the work data set already contains a copy of the RACF data set.
SYSUT1 DD	Defines a work data set on a direct access device. ICHUT200 requires this data set unless a RACF data set is used throughout processing.
SYSIN DD	Defines the control data set. The control data set is normally found in the input stream; however, it can be a member of a procedure library or a sequential data set existing elsewhere.
SYSUT2 DD	Defines a sequential message data set.
SYSPRINT DD	Defines a sequential data set for printed output. The data set can be written to an output device, a tape volume, or a direct access device.

Note: If the utility is run under TSO, you can allocate both the SYSIN and SYSUT2 data sets to the terminal. Although you can allocate SYSPRINT to the terminal, it should be allocated to SYSOUT because ICHUT200 may produce a large volume of output.

Utility Control Statements

ICHUT200 is controlled by utility control statements that have the following format:

```
INDEX [FORMAT]
I
```

where:

INDEX specifies you want the index scan function performed.

FORMAT specifies a formatted listing of all the index blocks.

Only one blank can separate INDEX and FORMAT.

You can use only columns 1 through 72.

```
MAP [ALL]
M
```

where:

MAP specifies you want the BAM/allocation verification performed.

ALL specifies that you want the encoded map for each BAM block in the RACF data set printed.

Only one blank can separate MAP and ALL.

You can use only columns 1 through 72.

```
END
```

where:

END terminates the utility program. An end-of-file on SYSIN also terminates the program.

You can use only columns 1 through 72.

ICHUT200 Example

In the following example, ICHUT200 copies a RACF data set to the SYSUT1 data set. A summary listing of all the index blocks is printed. Any BAM that contains conflicts is also printed with a table of the locations of the conflicts.

```
//VERIFY     JOB     USER=RAC01,PASSWORD=XXX
//STEP       EXEC    PGM=ICHUT200
//SYSRACF    DD      DSN=SYS1.RACF,DISP=SHR
//SYSUT1     DD      UNIT=SYSDA,SPACE=(CYL,(10)),
                     DCB=(LRECL=1024,RECFM=F)
//SYSUT2     DD      SYSOUT=A
//SYSPRINT   DD      SYSOUT=A
//SYSIN      DD      *
    INDEX
    MAP
    END
/*
```

ICHUT200 Unnumbered Messages

Following are the unnumbered ICHUT200 messages that might be printed out to a message data set. Because these messages are unnumbered, they are arranged in this section in alphabetical order by message text.

Note: See *RACF Messages and Codes* for the ICHUT200 numbered messages.

BAM BLOCK CHAIN FIELD IS BROKEN - MAP FUNCTION TERMINATED.

Explanation: While processing the chain of BAM blocks, ICHUT200 found a zero chain field in the BAM before all the blocks (the number contained in the ICB) were processed.

BAM BLOCK CHAIN FIELDS ARE IN A LOOP - MAP FUNCTION TERMINATED.

Explanation: ICHUT200 was processing the BAM when the count of the number of BAMs in the ICB was exceeded. The fields might be in a loop.

COUNT OF BAM BLOCKS IN ICB IS ZERO - MAP FUNCTION TERMINATED.

Explanation: The ICB contains a count of zero for the number of BAM blocks in the RACF data set.

COUNT OF NUMBER OF BLOCKS DEFINED BY A BAM IS INVALID - MAP FUNCTION TERMINATED.

Explanation: The count of the number of blocks defined by the BAM contained in the header is either zero or greater than 2008.

DATA BLOCK FAILED VALIDITY CHECK.

Explanation: The segment pointed to by a level one index block does not begin with the value X'82'.

DATA BLOCK KEY LENGTH INVALID.

Explanation: The record name in the profile is not from 1 to 44 bytes in length.

DISPLACEMENT TO FREE SPACE IS INCORRECT.

Explanation: The offset (in the header of the index block) to the free space in the block is incorrect, or the end-of-block delimiter (X'0C') is not present.

DISPLACEMENT TO LAST KEY IS INCORRECT.

Explanation: The offset (in the header of the index block) to the last entry is incorrect, or the entry identifier (X'21' or X'20') is not present.

E(P) BYTE/RBA OF NEXT BLOCK IN SEQUENCE SET IS INVALID.

Explanation: The sequence set pointer entry in the level one index block is not preceded by the value X'6x', or the next level one block is invalid for one of the following reasons:

- The first two bytes are not zero.
- The last four bytes are zero and this is not the last block in the chain.
- The RBA is not a multiple of 1024.

E(P) BYTE/RBA xxxxxxxxxxxx FAILED VALIDITY CHECK.

Explanation: The pointer entry of an index entry in the block is not preceded by the value X'6x', or the RBA xxxxxxxxxxxx of the next level index block or profile is invalid for one of the following reasons:

- The first two bytes are not zero.
- The last four bytes are zero.
- The RBA is not a multiple of 1024.
- For level one blocks, the RBA is not a multiple of 256.

ICHUT200 does not dump the index block if only the RBA is invalid.

END OF DATA FLAG BYTE POSSIBLY MISSING.

Explanation: The end-of-block delimiter at the end of the index block is not X'0C' or the displacement to this byte is incorrect. The displacement is calculated by adding the sum of the length of the last entry name in the block and the length of the pointer entry to the offset of the last entry name in the block. If the length of the entry name is incorrect, the displacement to this byte is incorrect.

FOLLOWING LEVEL 01 BLOCK IS NOT POINTED TO BY A LEVEL 02 BLOCK.

Explanation: An index block with a level greater then X'02' points to an index block with a level of X'01' in the header. ICHUT200 processes the level one index block normally.

I/O ERROR REREADING BAM BLOCK - MAP FUNCTION TERMINATED.

Explanation: ICHUT200 encountered an unrecoverable I/O error while attempting to reread a BAM block. The block is not dumped.

INDEX BLOCK FAILED VALIDITY CHECK.

Explanation: The block does not begin with the value X'8A'.

INVALID E(K) BYTE IN KEY ENTRY AT OFFSET aa

Explanation: An index entry name might not be preceded by a valid key byte. All entries in index blocks that are not level one must begin with the value X'21'. In level one index blocks, either X'22' or X'21' must precede each entry except for the last entry, which must be preceded by X'20'.

KEY ENTRY LENGTH INVALID AT OFFSET aa

Explanation: An index entry name does not have a valid length. An entry, other then the first entry in a block that is not level one, might have a zero length. If it does, it must also have a compression count other than zero. The compression count must not be greater then the length of the first entry in the block. The field 'aa' is the offset of the beginning of the invalid entry in the index block.

LOGICAL LENGTH OF DATA BLOCK IS INVALID.

Explanation: The logical length of the profile is not a multiple of 256, or is greater then the allocated length as defined in the header.

MORE THAN 200 BAM ALLOCATION ERRORS FOUND.

Explanation: In verifying the BAM blocks with the actual allocation of segments in the RACF data set, ICHUT200 found more then 200 locations with possible conflicts.

NON LEVEL 01 INDEX BLOCK IS IN SEQUENCE SET.

Explanation: The index block is in the sequence set, but the level in the header is not one.

POSSIBLE COMPRESSION COUNT ERROR IN KEY ENTRY AT OFFSET aa

Explanation: An index entry name might not have a valid compression count. The first entry must have a zero compression count. An entry, other then the first entry, must have a compression count which is less than or equal to the length of the first entry name. The field 'aa' is the offset of the beginning of the invalid entry in the index block.

POSSIBLE LOOP IN SEQUENCE SET.

Explanation: The first entry name of the level one index block is not alphabetically greater then the first entry name of the previous level one index block.

PROGRAM LIMIT EXCEEDED - PROCESSING OF INDEX BLOCKS TERMINATED.

Explanation: ICHUT200 found more then six levels of index blocks. Index block processing is terminated after six levels have been processed. Level one blocks are not processed.

RBA INVALID FOR TEMPLATE AT OFFSET xx RBA yyyyyyyyyyyy

Explanation: The RBA yyyyyyyyyyyy for the template defined at offset xx in the ICB is invalid for one of the following reasons:

- The first two bytes are not zero.
- The last four bytes are zero.
- The RBA is not a multiple of 1024.

RBA OF FIRST BAM BLOCK IS INVALID - MAP FUNCTION TERMINATED.

Explanation: ICHUT200 found an error in the RBA of the first BAM block (in the ICB). One of the following conditions caused the error.

- The last four bytes are zero.
- The first two bytes are not zero
- The last ten bits are not zero (denoting an address not on a 1K boundary).

RBA OF FIRST BLOCK OF INDEX SEQUENCE SET IS INVALID.

Explanation: The RBA of the first block of the index sequence set (in the ICB) is invalid for one of the following reasons:

- The first two bytes are not zero.
- The last four bytes are zero.
- The RBA is not a multiple of 1024.

RBA OF NEXT BAM BLOCK IS INVALID - MAP FUNCTION TERMINATED.

Explanation: The RBA of the next BAM block is invalid for one of the following reasons:

- The first two bytes are not zero.
- The last four bytes are zero and this is not the last BAM block in the chain.
- The RBA is not a multiple of 1024.

RBA OF TOP LEVEL INDEX BLOCK IS INVALID - PROCESSING TERMINATED.

Explanation: ICHUT200 found an error in the RBA of the top level index block (in the ICB). One of the following conditions caused the error:

- The last four bytes are zero.
- The first two bytes are not zero.
- The last ten bits are not zero (denoting an address not on a 1K boundary).

READ FAILED FOR TOP LEVEL INDEX BLOCK - PROCESSING TERMINATED.

Explanation: ICHUT200 encountered a permanent I/O error while attempting to read the top-level index block. The block is not dumped.

SEQUENCE SET CHAIN FIELD IS BROKEN.

Explanation: When processing all the index blocks, ICHUT200 keeps a count of the level one blocks. ICHUT200 uses this count while processing the chain of level one blocks (sequence set). While following the chain of level one blocks, the utility program found a zero sequence set RBA before it reached the count of level one blocks.

TEMPLATE COUNT IN ICB IS INVALID.

Explanation: The ICB contains a count of the number of templates that is either zero or greater than the number of spaces allocated for template definitions.

TOP LEVEL INDEX BLOCK FAILED VALIDITY CHECK - PROCESSING TERMINATED.

Explanation: The top-level index block, pointed to by the ICB, does not begin with the value X'8A'.

Block Update (BLKUPD) Utility Command (ICHUT300)

ICHUT300 is a RACF utility command used to modify any block on a RACF data set. The utility is invoked by the RACF command BLKUPD. There are no utility control statements; job control statements are not required.

The following commands control the functions of ICHUT300 after it is invoked by the BLKUPD command:

- READ command - reads a RACF data set block into storage. Subcommands of the READ command update the block.

- LOCATE command -- locates an index entry in the RACF sequence set.

- END command (required) - terminates the processing of ICHUT300.

Considerations When Using ICHUT300

The following considerations apply to the use of the block update utility:

- ICHUT300 does not support attention handling. If an attention interruption occurs, and the user enters any data other than a null line or TIME command, the BLKUPD command and its subcommands terminate.

- When executing ICHUT300 under control of command procedures, you must add the DATA and ENDDATA command procedure statements around each ICHUT300 command and subcommand that is also a command procedure statement. See the *TSO Command Language Reference* for a description of command procedures.

- When executing ICHUT300 under a TMP (terminal monitor program) that allows multitasking, you cannot have any other active task in your session. Allow ICHUT300 to complete before executing any other TSO command.

- You can use ICHUT300 on any active RACF data set or a copy of the RACF data set.

- The subcommands of BLKUPD follow the TSO syntax rules. See the *TSO Command Language Reference* for a description of TSO command syntax rules.

- If a RACF data set is RACF-protected, you must have at least UPDATE authority to the data set.

- ICHUT300 runs as an APF-authorized program. Because any changes made to the data set are not made to the in-storage blocks, do not use ICHUT300 to make changes against an active RACF-protected data set if you are using the in-storage data or index block options.

Invoking ICHUT300

The BLKUPD command invokes ICHUT300. Optionally, this command can allocate a RACF data set.

The format of the BLKUPD command is as follows:

```
BLKUPD [racfdsname]
```

where:

racfdsname specifies the name of a cataloged RACF data set. The name can be from 1 to 44 characters. If a name is not specified, you must have allocated the RACF data set on a SYSRACF DD statement before invoking ICHUT300.

Updating a RACF Block

You can use the READ function to update a RACF data set block. The READ command causes a block to be read into storage and copied into a work area. The subcommands of the READ command update the block in the work area or print portions of the original block or the updated block. The updated block is called the new block; the original block is called the old block.

The format of the READ command is as follows:

```
READ rba UPDATE
         NOUPDATE
```

where:

rba specifies the relative byte address of the block to be read. If rba does not represent an address on a 1K boundary, the block on the 1K boundary that contains rba is read. You can enter the value rba as a hexadecimal (X'nn') or decimal (nnn) number. Hexadecimal numbers can be from 1 to 8 characters; decimal numbers can be from 1 to 10 characters.

UPDATE specifies that this RACF data set is to be under exclusive control until the READ function terminates. You should use UPDATE only during slack time in system operations..

NOUPDATE specifies that this RACF data set is for shared use only. Commands that update the block are rejected. NOUPDATE is the default.

The following subcommands control the READ function:

- LIST subcommand - prints portions of the block in hexadecimal
- FORMAT subcommand - produces a formatted listing of the contents of an index block
- FIND subcommand - locates the offset of data in the block
- REP subcommand - replaces data in the block
- DISPLAY subcommand - displays the entries of a new index block. Subcommands of DISPLAY update the entries.
- REREAD subcommand - overlays the new block in the work area with the old block
- END subcommand - terminates the READ function

LIST Subcommand

The LIST subcommand prints a hexadecimal dump of all or part of the RACF block.

The format of the LIST subcommand is as follows:

```
LIST OLD RANGE(xxx,yyy)
     NEW ALL
```

where:

OLD specifies the original block obtained by the READ command. OLD is the default.

NEW specifies the block that was updated.

RANGE defines the parts of the block to be listed.

xxx specifies the offset into the block where the listing is to start. The default is 0.

yyy specifies the number of bytes to be listed. The default is 16.

The values for xxx and yyy can be hexadecimal or decimal numbers. If the sum of xxx and yyy exceeds 1024, the dump ends at the end of the block.

ALL specifies a hexadecimal listing of the entire RACF block.

FORMAT Subcommand

The FORMAT subcommand prints a formatted list of the contents of an index block. The list contains the following information:

- The RBA of the block
- The level of the block
- The offset to the last entry
- The offset to free space
- The offset of each entry within the block
- The front-end compression count of each entry
- The name of each entry (generic names are identified by a G in parentheses after the name)
- The last qualifier encode byte of each entry
- The RBA of the next-level index block for each entry or, for level one blocks, the RBA of the profile

If the block obtained by the READ command is not a valid index block, the request is rejected.

The format of the FORMAT subcommand is as follows:

```
FORMAT OLD
FMT    NEW
```

where:

OLD specifies the original index block obtained by READ. OLD is the default.

NEW specifies the index block that was updated.

FIND Subcommand

The FIND subcommand finds and prints the hexadecimal offset of a string in the block.

The format of the FIND subcommand is as follows:

```
FIND string OLD
            NEW
```

where:

string specifies the string to be located. The string can be hexadecimal (which is specified as X'string') or characters (which can be quoted or unquoted). Hexadecimal strings are right-justified and padded with zeroes to a byte boundary.

OLD specifies the original block obtained by READ. OLD is the default.

NEW specifies the block that was updated.

REP Subcommand

The REP subcommand replaces a string in the new block with another string. Optionally, this subcommand verifies the original string before making the replacement.

The format of the REP subcommand is as follows:

```
REP string  OFFSET(xxx) [VER(string)]
```

where:

string specifies the new string of data. String can be hexadecimal (which is specified as X'string') or characters (which can be quoted or unquoted). Hexadecimal strings are right-justified and padded with zeroes to a byte boundary.

OFFSET(xxx) specifies the offset into the block where the string is to be replaced.

VER(string) specifies the original string that is to be verified. If the string is not found at offset xxx into the new block, the replacement is not made.

DISPLAY Subcommand

The DISPLAY subcommand displays each entry of a new index block. The subcommands of DISPLAY are used to modify the displayed entries. If the block obtained by the READ command is not a valid index block, the request is rejected.

Note: Entry names for general resources must begin with the first 4 characters of the classname followed by a dash (for example, DASD-). Unless space in the index blocks is a problem, enter the entry name as is. RACF performs compression when the index block becomes full.

The format of the DISPLAY subcommand is as follows:

```
DISPLAY  [ENTRY(name)]  [GENERIC]
```

where:

ENTRY(name) specifies the index entry where the display begins. If you do not specify a name, the display begins at the first entry in the index block; name can be from 1 to 44 alphanumeric characters.

GENERIC: indicates that the index specified in ENTITY(name) is a generic name.

The subcommands of DISPLAY are as follows:

```
NEXT
N
```

which displays the next entry. The displayed entry is not changed. A null line also displays the next entry.

```
DELETE
D
```

which deletes the index entry. The next entry is displayed.

```
CHANGE [ENTRY(name)] [GENERIC] [RBA(rba)]  [COMP(xx)] DUPLICATE
C                                                     NODUPLICATE
```

which changes the name, the RBA of the index entry displayed, the compression count, and/or the entry identifier byte. ICHUT300 displays the changed entry; you must enter the NEXT subcommand or a null line to display the next entry.

Some definitions and considerations are:

- The values xx and rba can be from 1 to 8 hexadecimal characters or from 1 to 10 decimal characters.
- The length of name plus the compression count cannot exceed 44. To create a fully compressed name, specify ENTRY (' ') with a nonzero compression count. (If the first entry in the index block is encoded, do not specify full compression; compress only up to the last period (.) in the entry name or ignore compression.)
- DUPLICATE changes the entry identifier byte to X'22'; NODUPLICATE changes the byte to X'21'. If you omit both DUPLICATE and NODUPLICATE, the entry identifier byte is not changed.
- GENERIC identifies the entry name as a generic name.

```
INSERT ENTRY(name) [GENERIC] RBA(rba) [COMP(xx)] DUPLICATE
I                                                NODUPLICATE
```

which inserts a new index entry immediately preceding the entry that is displayed. ICHUT300 displays the new entry; entering the NEXT subcommand displays the original entry again.

Some definitions and considerations are:

- GENERIC indicates that the index entry specified in ENTRY(name) is a generic name.
- The values xx and rba can be from 1 to 8 hexadecimal characters or from 1 to 10 decimal characters.
- The length of name plus the compression count cannot exceed 44. To create a fully compressed name, specify ENTRY (' ') with a nonzero compression count. (If the first entry in the index block is encoded, do not specify full compression; compress only up to the last period (.) in the entry name or ignore compression.)
- DUPLICATE sets the entry identifier byte to X'22'; NODUPLICATE sets the byte to X'21'. NODUPLICATE is the default.

```
END       SAVE
          NOSAVE
```

which ends the DISPLAY function. You can continue to update the block under the READ function.

SAVE updates the new index block to reflect the changes made under the DISPLAY subcommand. **NOSAVE** ignores all changes made under the DISPLAY subcommand. If UPDATE was not specified on the READ command, NOSAVE is forced.

REREAD Subcommand

The REREAD subcommand overlays the new RACF block in the work area with the old block (in case mistakes were made in updating the new block).

The format of the REREAD subcommand is as follows:

```
REREAD
```

END Subcommand

The END subcommand ends processing on the RACF data set block obtained by the READ command.

The format of the END subcommand is as follows:

```
END       SAVE
          NOSAVE
```

where:

SAVE specifies that the new block is written back to the RACF data set in place of the one that was read.

NOSAVE specifies that the RACF data set is not updated with the new block. NOSAVE is forced if UPDATE was not specified on the READ command or if no changes were made in the block.

Locating an Index Entry

The LOCATE command locates an index entry in the sequence set and prints a formatted list of the contents of the level one index block containing the entry name. Optionally, LOCATE prints a listing of all the index blocks in the chain from the highest-level index block to the appropriate level one block. The formatted list contains the same information as that produced by the FORMAT subcommand of the READ command.

If an index block in the chain contains an error, ICHUT300 produces a hexadecimal dump of the block. If the name is not found in a level one block, ICHUT300 prints the block that should contain it. If the block containing the name is not found by a hierarchy index search because of errors in the chain, ICHUT300 searches the sequence set.

The format of the LOCATE command is as follows:

```
LOCATE entryname    [GENERIC]    [LISTALL]
```

where:

entryname specifies the index entry to be located; entryname can be from 1 to 44 characters. Entry names for general resources must begin with the first 4 characters of the class name followed by a dash (for example, DASD-).

GENERIC specifies that entryname is generic.

LISTALL specifies that all the index blocks in the hierarchy chain are to be printed.

Terminating ICHUT300

The END command terminates ICHUT300.

The format of the END command is:

```
END
```

Locating and Entering Connect Entries

A connect entry in an index block is composed of a userid and a group name separated by X'00'. This X'00' is unprintable. Therefore, when the FORMAT or DISPLAY subcommand formats a connect entry, they substitute a slash (/) in the output for the X'00'. The following restrictions apply to ICHUT300 commands when locating and entering connect entries:

- When using the CHANGE or INSERT subcommand, substitute a printable character (such as /) for X'00' in the connect entry name on the ENTRY keyword. After you have ended the DISPLAY function, use the REP subcommand to replace the character with X'00'.

- When using the LOCATE command to locate the level one index block containing a connect entry, enter just the userid from the connect entry. This entry is before the connect entry in the sequence set and will probably be in the same index block as the connect entry.

- You cannot enter a connect entry name in the ENTRY keyword of the DISPLAY command. Specify another index entry in the block or allow the display to begin with the first entry.

ICHUT300 Example

Following are examples of commands that you might issue during a single execution of ICHUT300.

```
BLKUPD 'SYS1.RACF'
```

invokes the utility, specifying the RACF data set to be allocated.

```
READ X'19400' UPDATE
```

reads the block at RBA 19400 and specifies that changes will be made to the block.

```
FORMAT
```

prints a formatted listing of the old index block obtained by the READ command.

```
FIND M267
```

locates the offset of the character string 'M267' in the old block.

```
REP M270 OFFSET(X'42') VER(M267)
```

verifies that offset X'42' in the new block contains the character string 'M267' and replaces that string with the character string 'M270'.

```
LIST NEW RANGE(X'42',8)
```

lists the modified area of the new block (8 bytes beginning at offset X'42').

```
DISPLAY
```

displays the first entry in the updated index block.

```
DELETE
```

deletes the index entry just displayed. The next entry, which moves up to replace the deleted one, is displayed.

```
INSERT ENTRY(X200ABC) RBA(X'19800')
```

inserts a new index entry, with the specified name and RBA, immediately preceding the entry just displayed. The new entry is displayed.

```
NEXT
```

displays the original entry with the new offset.

```
CHANGE ENTRY(X550ABC.A)
```

changes the entry name of the entry just displayed to 'X550ABC.A'. The compression count and the RBA are unchanged.

```
END SAVE
```

ends the DISPLAY command and saves the changes that were made in the index block. The utility still executes under the READ command.

```
FORMAT NEW
```

prints a formatted listing of the updated index block.

```
END SAVE
```

ends the READ command and writes the new block out to the RACF data set.

```
LOCATE SALES.* GENERIC
```

prints a formatted list of the level one block that contains the generic entry name 'SALES.*'.

```
END
```

terminates ICHUT300.

The RACF Data Set Split/Merge/Extend Utility Program (ICHUT400)

ICHUT400 is a RACF utility program that supports multiple RACF data sets. It performs the following functions:

- Copies a RACF data set to a larger or smaller data set, provided there is enough space for the copy.
- Redistributes data from a RACF data set(s). For example, ICHUT400 can split a single RACF data set into multiple RACF data sets, merge multiple RACF data sets into fewer RACF data sets, or rearrange RACF profiles across the same number of input and output RACF data sets. The utility allows a maximum of 255 input data sets and 255 output data sets.
- Identifies inconsistencies, such as duplicate resource names appearing in different classes.

Considerations When Using ICHUT400

ICHUT400 formats and initializes each output data set with an ICB, templates, BAM blocks, a complete index structure, and profiles from the input data set(s). It also provides the following features:

- **Index compression:** ICHUT400 builds the index structure of each output data set from the lowest-level upwards. Because no block splitting occurs, the index structure is automatically compressed. You can specify a percentage of free space to be left in each index block.
- **Block alignment:** You can request that RACF attempt to keep any profile that is not larger than one block (1K) within a block boundary.
- **Index structure correction:** The utility uses only the sequence set blocks of the input data sets; it builds the rest of the index structure from the sequence set. As a result, inconsistencies in higher-level index blocks are corrected and do not prevent the utility from executing correctly.
- **Multiple input data sets:** When running with more than one input data set, ICHUT400 copies from the lowest number INDDn data set.

Extending a Data Set Using ICHUT400

Use the split/merge/extend utility to copy an existing RACF data set to a larger data set, even if you are not splitting or merging the data set(s).

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

Executing the Split/Merge/Extend Utility

The following job control statements are necessary for executing ICHUT400:

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM = ICHUT400) or, if the job control statements are in a procedure library, the procedure name. You can also request ICHUT400 processing options by specifying parameters in the PARM field. See "Parameter Specification."
SYSPRINT DD	Defines a sequential message data set. The data set can be written to an output device, a tape volume, or a direct access volume.
INDDn DD	Defines the RACF input data set(s). See "Input Data Set Specification."
OUTDDn DD	Defines the RACF output data set(s). This statement is not required if you are executing the utility only to identify inconsistencies in a RACF data set(s). See "Output Data Set Specification."

If you are redistributing the profiles across more than one RACF data set, you must also provide a *range table.* The range table indicates which profiles are placed in each output data set. See "Range Table" in Chapter 5 and "Output Data Set Selection" later in this chapter. If any of the input data sets are RACF-protected, you must have at least UPDATE authority for those data sets.

Input Data Set Specification

Allowable ddnames for input data sets are INDD1 through INDD255. The input data sets must be numbered consecutively. For example, if 25 input data sets are provided, they must be assigned ddnames INDD1 through INDD25. The utility processes the input data sets until a number is omitted. You must provide at least one input data set (INDD1).

The only considerations in ordering the input data sets (that is, which data set you assign to INDD1, which to INDD2, and so on) are the following:

- The utility copies the ICB and templates from the input data set defined by INDD1 to all output data sets.
- If you do not allow duplicate names in the DATASET class or if duplicate entries exist within any other class, the utility copies the entry from the input data set identified by the lowest-numbered ddname.

You can specify active RACF data sets as input data sets. However, to prevent updates to the data sets during the processing of the utility, you should specify the LOCKINPUT option. (See "Parameter Specifications"). Each input data set must be a correctly formatted RACF data set.

| *Note:* The ICHUT400 utility does not copy empty RACF data sets; the index must contain valid profile data set entries for the utility to copy the data set.

Output Data Set Specification

When redistributing or copying a RACF data set, you must code an OUTDDn statement for every RACF data set that the utility will create. The ddnames of the statements defining the output data sets have a relationship to the range table. If a range table is provided, ICHUT400 uses the greatest data set number in the table as an upper boundary for processing. If no range table is provided, the upper boundary is one. ICHUT400 does not process any output data set identified by a ddname with a number greater than the upper boundary. You can specify as many as 255 output data sets on statements named OUTDD1 through OUTDD255.

Output data sets can be new or old direct access data sets. As with all RACF data sets, RACF uses only the first extent. Therefore, do not create any data sets with a secondary space allocation.

If you are executing ICHUT400 only to identify inconsistencies in the RACF data set(s), do not code an OUTDDn statement. See "Processing of Conflicts and Inconsistencies" for a description of inconsistencies found in the data set(s).

Output Data Set Selection

If multiple output data sets are created, the utility uses the range table to determine which profiles to copy to which output data sets. Therefore, you must supply a range table that indicates the range of profiles to be placed on each data set. You specify the name of the module that contains the range table in the TABLE parameter of the PARM field in the EXEC statement. (See "Parameter Specification.") Chapter 5 describes the format of the range table. If you are using a new range table, you should check that the data set name table is consistent with the new range table. See Chapter 5 for a description of the data set name table.

Output Data Set Processing

ICHUT400 initializes each provided output data set as a completely independent RACF data set, with an ICB, templates, BAM blocks to describe free space, and an index structure to describe the data set's contents.

ICHUT400 processing is similar to the processing of the ICHMIN00 utility. As does ICHMIN00, ICHUT400 uses only the first extent of the data set. For old data sets, the user must ensure that only one extent is currently allocated because the utility cannot detect multiple extents for existing data sets.

The two utilities are different, however, because ICHMIN00 builds the templates using control records read from the SYSTEMP file; ICHUT400 merely copies templates read from INDD1. Also, ICHMIN00 builds an ICB with default option settings; ICHUT400 copies the option settings from the ICB of INDD1.

ICHUT400 builds the index of each output data set sequentially from the bottom up. You can request that free space be left in each index block by specifying the FREESPACE parameter. (See "Parameter Specification" later in this chapter). Specifying FREESPACE allows new entries to be added to a data set while RACF is active, without causing an index block split.

Aside from the free space requested, the utility compresses the index. ICHUT400 also writes profiles to the output data sets in collating sequence. You can request, with the ALIGN parameter, that no profile span physical blocks if it can fit into a single physical block (1024 bytes). This decreases the amount of I/O required to access profiles that occupy two, three, or four 256-byte segments. The option has no effect on profiles that occupy only one segment.

Parameter Specification

You can specify a number of parameters in the PARM field of the EXEC statement of the step executing ICHUT400. The syntax for the parameters is similar to that of the TSO command language. They can be separated by one or more blanks. Embedded blanks are not allowed. Any keyword can be abbreviated to the number of initial characters that uniquely identify that keyword. The specification of redundant or contradictory keywords is considered an error. The allowable keywords are:

LOCKINPUT/NOLOCKINPUT
Provides a means of ensuring that, with the exception of updating statistics, any RACF data sets used as input are not updated. Specifying LOCKINPUT means updates are no longer allowed to an input data set for which LOCKINPUT is specified, even after the utility terminates. To indicate that updates are no longer allowed, the utility sets the extend bit (at offset 26 in the ICB) on. NOLOCKINPUT is the default.

TABLE(table-name)/NOTABLE
Permits the specification of a user-written range table to be used to select an output data set for each profile. Specifying TABLE(table-name) indicates that the named load module is to be used. NOTABLE is the default; either specifying or defaulting to it forces the selection of OUTDD1 for all profiles.

FREESPACE(percent)/NOFREESPACE
Allows you to control the amount of free space left in index blocks created for the output data sets. You can specify that from 0 to 50 percent of the space within the index block is to be left free. The sequence set (level one) will contain the specified percentage of free space; level two will contain one seventh of the specified percentage. Index levels higher than two will contain no free space. NOFREESPACE (which is equivalent to FREESPACE(0)) is the default.

ALIGN/NOALIGN
Allows you to control profile space allocation. Specifying ALIGN forces profiles that occupy two, three, or four 256-byte segments to be placed so that they do not span 1024-byte physical blocks. Having a single physical block can decrease the I/O needed to process these profiles. Specifying NOALIGN (the default) causes no special alignment.

DUPDATASETS/NODUPDATASETS

Allows you to control the processing of DATASET entries with identical names from different input data sets. Specifying DUPDATASETS indicates that duplicates are allowed and that all DATASET entries are to be processed. If you specify NODUPDATASETS and the utility encounters duplicate entries on different data sets, the utility copies the DATASET entry from the input data set identified by the lowest-numbered ddname. When NODUPDATASETS is in effect, duplicates occurring on a single input data set are all accepted, assuming that they do not conflict with an entry from another data set earlier in the selection sequence. NODUPDATASETS is the default.

Processing of Conflicts and Inconsistencies

If more than one input data set is being processed, you can encounter entries with duplicate names. The way in which entries with duplicate names are processed depends on which classes they belong to. If the utility encounters duplicate names across classes or within classes other than DATASET, it copies the entry from the input data set identified by the lowest-numbered ddname and issues a message indicating the entry that was not copied. See the description of the DUPDATASETS/NODUPDATASETS parameter for information on how duplicate DATASET entries are handled.

If there is a logical error in an input RACF data set, it is possible that an index entry does not point to the profile for the correct entity. In this case, ICHUT400 issues an error message because it is impossible to copy the entity to an output data set.

The possibility of conflicts between tape volume sets exists even when only one input data set is specified. The utility detects:

- If more than one output data set is specified, a tape volume set might contain members that are assigned to different output data sets by the range table. Because of the way that tape volume sets are implemented, it is impossible to reconstruct such a tape volume set on the output data sets. Therefore, ICHUT400 does not copy the entire tape volume set, including all its members, and issues a message to that effect.

- It is possible for two tape volume sets to contain one or more members with the same names. Usually, this happens only when more than one input data set is specified. (Because of an internal inconsistency however, it is possible for it to happen even with only one input data set.) It is impossible for ICHUT400 to copy both of these tape volume sets to the output data set. A message is issued to indicate which data set is not copied.

Utility 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
8 (8)	One or more error conditions occurred because of one of the following reasons: ● 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 data set
10 (16)	A terminating error condition occurred because of one of the following reasons: ● A recovery environment could not be established ● The SYSPRINT file could not be opened ● An error was found in a parameter specification ● A range table was requested but could not be loaded ● An error was detected in the specified range table. ● An error occurred on an input data set

ICHUT400 Examples

These examples show how to code the split/merge/extend utility (ICHUT400) to perform different functions.

Example 1

In this example, ICHUT400 copies the profiles from a single input data set to a single output data set. This one-to-one copy does not require a range table because all profiles are directed to OUTDD1. ICHUT400 compresses the output data set, which might be larger or smaller than the input data set, and might even reside on a different type of device. The index blocks of the output data sets will contain free space to allow for expansion.

```
//J1       JOB  USER=RACUSER,PASSWORD=XXX
//         EXEC PGM=ICHUT400,PARM='FREESPACE(20)'
//SYSPRINT  DD  SYSOUT=A
//INDD1     DD  DSN=SYS1.RACF5,DISP=OLD
//OUTDD1    DD  DSN=SYS2.RACF5,DISP=(,KEEP),
//              VOL=SER=333305,
//              SPACE=(CYL,10,,CONTIG),
//              UNIT=3330
```

Example 2

In this example, ICHUT400 splits a single RACF data set into three output data sets. ICHUT400 assigns profiles to output data sets using a range table in a module named SELECT. The load module resides in library INSTALL.LINKLIB.

```
//J2      JOB   USER=RACUSER,PASSWORD=XXX
//        EXEC  PGM=ICHUT400,PARM='TABLE(SELECT)'
//SYSPRINT  DD  SYSOUT=A
//INDD1     DD  DSN=SYS1.RACF,DISP=OLD
//OUTDD1    DD  DSN=SYS2.RACF1,DISP=(,KEEP),
//              UNIT=3330,VOL=SER=333301,
//              SPACE=(CYL,5,,CONTIG)
//OUTDD2    DD  DSN=SYS2.RACF2,DISP=(,KEEP),
//              UNIT=2314,VOL=SER=231402,
//              SPACE=(CYL,20,,CONTIG)
//OUTDD3    DD  DSN=SYS2.RACF3,DISP=(,KEEP),
//              UNIT=2314,VOL=SER=231403,
//              SPACE=(CYL,5,,CONTIG)
//STEPLIB   DD  DSN=INSTALL.LINKLIB,DISP=SHR
```

Example 3

In this example, ICHUT400 merges the RACF data sets from two different systems into a single RACF data set. ICHUT400 first makes a test run to identify any possible inconsistencies. Data set entries with identical names, but from different RACF data sets, are allowed.

```
//J3A       JOB    USER=RACUSER,PASSWORD=XXX
//          EXEC   PGM=ICHUT400,PARM='DUPDATASETS'
//SYSPRINT    DD   SYSOUT=A
//INDD1       DD   DSN=SYS1.RACF1,DISP=OLD
//INDD2       DD   DSN=SYS1.RACF2,DISP=OLD
```

After any identified inconsistencies are corrected, ICHUT400 performs the actual merge. To improve I/O performance, the utility is to align profiles written to the output data set.

```
//J3B     JOB
//        EXEC     PGM=ICHUT400,PARM=('DUPDA FREE(10)',
//                 ALIGN)
//SYSPRINT       DD SYSOUT=A
//INDD1          DD DSN=SYS1.RACF1,DISP=OLD
//INDD2          DD DSN=SYS1.RACF2,DISP=OLD
//OUTDD1         DD DSN=SYS2.RACF,DISP=(,KEEP),
//               UNIT=3330,VOL=SER=333301,
//               SPACE=(CYL,10,,CONTIG)
```

Example 4

The active RACF data set is full and requests are failing due to lack of space. In this example, ICHUT400 copies the RACF data set to a larger data set, rebalances the index structure, and provides room in each index block for expansion. ICHUT400 is to align the profiles to improve access time for some of the larger profiles. Because LOCK is specified for PARM on the EXEC statement in this example, no new entries can be made until you unlock the data set by resetting the lock bit in the inventory control block (ICB).

```
//J4      JOB
//        EXEC   PGM=ICHUT400,PARM='LOCK,F(10),A'
//SYSPRINT    DD SYSOUT=A
//INDD1       DD DSN=SYS1.RACF,DISP=OLD
//OUTDD1      DD DSN=SYS2.RACF,DISP=(,KEEP),
//               UNIT=3330,VOL=SER=333301,
//               SPACE=(CYL,15,,CONTIG)
```

Chapter 7. RACF Installation Exits

Installation Exits

RACF provides a number of installation exits where you can use your own routines to enhance the facilities offered by RACF as well as to optimize its usability.

The major functions within RACF have exits. There are exits for:

- User identification and verification
- Access checking
- Automatic profile creation and deletion
- Resource, user, and group profile manipulation commands

The RACF initialization routine, ICHSEC00, loads the exit routines during master scheduler initialization and places the exit addresses in the RACF communication vector table (RCVT). If RACF determines that the exit routines were not supplied (by a search of the LPA), the RCVT fields pointing to the exit routines are set to zero.

The exit routines must be re-enterable and refreshable and must be located in the link pack area (PLPA, FLPA or MLPA). The exit routines receive control with standard linkage conventions; the exit routines should use standard linkage conventions to return control. The exit routines receive control from:

- The RACINIT, RACHECK, RACDEF and RACLIST SVCs in supervisor state, under protection key zero, and with no locks held.

- The RACF manager in supervisor state and under protection key zero.

- The commands and utilities in problem state, under a user key, with no locks held, and under APF-authorization (this is the state in which the commands are executed).

- The FRACHECK routine in an environment offering limited function. Because FRACHECK is intended for high performance environments:

  - The exits are not provided a save area

  - The execution key is unpredictable

  - The exits might receive control in supervisor or problem program state

- The exits might or might not be given APF-authorized control
- The exits should not use any SVCs
- Locks might or might not be held on entry to the exits
- The exits must be reentrant and, if used with IMS, must reside in the fixed link pack area.

See Chapter 12 for a mapping of the ACEE (accessor environment element) data area, which is helpful when you code exit routines.

RACF supports callers residing above 16 megabytes, as well as permitting them to have parameters and parameter lists above the line. In an MVS/XA system RACF automatically runs in 31-bit mode and checks whether its callers are in 24-bit or 31-bit mode.

During RACF initialization, ICHSEC00 records the AMODE of each installation exit routine it finds. All exit routines are called in their defined AMODE and must return control to RACF in the mode in which they were called. All parameters and parameter lists passed to any exit routine are located below 16 megabytes.

Note: In many cases RACF must copy caller-supplied parameter areas to an area below 16 megabytes, so that the exit may address the parameter areas. However, when the RACHECK or RACDEF exit routines receive the ACCLVL parameter or INSTL parameters, RACF does not know the format or length of the parameters being passed, and therefore cannot copy the parameters into 24-bit storage. Because the parameter list pointing to these parameters is in 24-bit storage, you must modify the exit routines to handle the parameters if the exit routines access these areas, and if they are passed by callers in 31-bit mode.

Some RACF data areas exist above 16 megabytes. RACF exit routines with an AMODE of 24 cannot support the placement of these data areas above 16 megabytes.

RACDEF, RACHECK, RACINIT, RACLIST, and FRACHECK Exits

The RACDEF, RACHECK, and RACINIT preprocessing and postprocessing exits allow an installation to tailor the parameters passed on those macro instructions and to perform any additional security checks or processing that the installation requires. The RACLIST pre/postprocessing exit and selection exit allow the installation to modify RACLIST processing options and to resolve conflicts between new and existing profile information. The FRACHECK preprocessing exit allows the installation to make additional security checks or to instruct FRACHECK to either accept or fail a request.

The purpose of each SVC and possible uses of the exits follow:

- RACDEF SVC - used to define, modify, and delete discrete and/or generic DASD data set profiles, or profiles for any resource defined by classes in the class descriptor table.

The RACDEF exits can be used to cause all allocation authorization checks (DEFINE requests from DADSM ALLOCATE) to be accepted, regardless of the user's authority.

- RACHECK SVC - used to determine if a user is authorized to obtain use of a resource (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).

  Resource managers, such as catalog management, might use the RACHECK SVC to determine if a resource (including DATASET) is RACF-protected.

- RACINIT SVC - used to determine if a userid is defined to RACF and if the user has supplied a valid password and group name. During TSO logon processing, RACINIT also determines if a user who is entering the system has supplied a valid operator identification card and is authorized to access the terminal. During IMS/VS SIGN ON and CICS/VS CSSN (sign-on) processing, RACINIT determines if a user who is entering the system is authorized to use IMS/VS or CICS/VS and to access the terminal.

  If the userid, password, operator identification card, group name, and terminal are accepted, RACF builds an access environment element (ACEE) for the user. (Note that, when no userid, group, and password are passed to RACINIT, RACINIT builds a default ACEE containing an '*' (X'5C') for the userid 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 or batch job. You can use the RACINIT exit routine to supply a userid for undefined users or to perform additional authorization checks for users.

- RACLIST SVC - used to build in-storage (resident) copies of general resource profiles. Both the RACHECK SVC and FRACHECK routine use these resident profiles for authorization checking.

  RACLIST processing is as follows:

  1. RACF calls the pre/postprocessing exit routine to perform initialization of the installation environment.

  2. If the resource class being processed has a resource group class associated with it, then for every entity in the resource group class:

     a. RACLIST processes each member in the resource group entity individually.

     b. RACLIST calls the selection exit routine to resolve conflicts between the information associated with the member resource currently being processed and a previously-built profile for that member. (This situation will exist if a resource is a member of more than one grouping entity.)

   c. RACLIST builds an in-storage profile for the member resource (or updates the previously-built profile).

3. For each resource in the class (or specified by the LIST option):

   a. RACLIST calls the selection exit routine to resolve conflicts between the information associated with the resource currently being processed and a previously-built profile for that resource. (This situation will exist if a resource has an individual profile in a RACF data set and is a member of one or more resource group entities.)

   b. RACLIST builds an in-storage profile for the resource (or updates the previously-built profile).

4. RACF calls the pre/postprocessing exit routine to cleanup the installation environment.

Note: The FRACHECK routine is not an SVC and is used only to check authorizations to resources protected by RACLIST-created profiles.

Exit Requirements

Requirements for these exits are:

- The RACDEF preprocessing exit routine must be named ICHRDX01. It is entered before the definition, modification, or deletion of resource profiles.

- The RACDEF 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 data set.

- The RACHECK preprocessing exit routine must be named ICHRCX01. It is entered before authorization checking of resources.

- The RACHECK postprocessing exit routine must be named ICHRCX02. It is entered after authorization checking of resources.

- The RACINIT preprocessing exit routine must be named ICHRIX01. It is entered before user identification and verification and terminal authorization checking.

- The RACINIT postprocessing exit routine must be named ICHRIX02. It is entered after user identification and verification, and terminal authorization checking.

- The RACLIST pre/postprocessing exit must be named ICHRLX01. It 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).

- The RACLIST selection exit must be named ICHRLX02. It is entered as each profile is being built.

  A resource name can appear in more than one resource group profile and at the same time can have a profile of its own. RACLIST resolves conflicts between these multiple profiles for the following fields:

  - UACC
  - LEVEL
  - Audit options
  - Global audit options
  - Installation data
  - Access list entries
  - Owner

  The RACLIST pre/postprocessing 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. The RACLIST selection exit can also resolve conflicts for the OWNER field.

- The FRACHECK preprocessing exit must be named ICHRFX01. It is entered before the FRACHECK service routine performs authorization checking.

- 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.

On entry to the exit routines, register 1 points to an area containing the parameters specified on the macro instruction. The information supplied on the INSTLN parameter is also available at this time.

When the preprocessing exit routines receive control, RACF has already validity-checked the macro instruction parameters, but has not yet performed any other SVC processing. However, if the installation has included a started procedure name in the installation's started procedures module, RACINIT will have already converted the started procedure name to a userid and, optionally, a group name.

When the RACHECK and RACINIT postprocessing exit routines receive control, RACF has already performed the main SVC function (for example, authorization checking and ACEE creation), but has not performed any logging or statistics recording.

When the RACHECK and RACDEF preprocessing exits receive control for the DATASET class, RACF has already processed the naming conventions table, if there is one.

Note: Make changes or additions to the parameter information only in the designated areas.

Password Encryption Exit

The encryption exit routine ICHDEX01 is called by the RACF manager whenever it is necessary to store or compare encrypted password or OIDCARD data in a user profile. RACF may have enqueued on the RACF data set 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.

RACF has a dummy encryption exit routine, ICHDEX01, that unconditionally returns with a return code of 4, to force RACF to use the masking algorithm for password and OIDCARD data.

You can use this exit routine to:

- Replace the data encryption standard (DES) algorithm that encrypts passwords and OIDCARD data with any algorithm that you want (including the masking algorithm).

- Always set a return code of 8, instructing RACF to ignore the old masking algorithm and only use DES processing.

New Password Exit

The RACINIT SVC and the ALTUSER and PASSWORD TSO commands invoke the 'new password' processing exit, which must be named ICHPWX01. This exit is entered when a new password or password change interval is to be established; it is not entered when the password is set to the default group name.

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 gains control if the following conditions are true:

- The new password is not a duplicate of the current password

- The new password is not a duplicate of a previous password if the password history option is active

- The new password follows all of the installation's syntax rules

Naming Convention Table

RACF requires a data set name format where the high-level qualifier of a data set name is a RACF-defined userid 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 SVC routines 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 "ICHNCONV Macro" in Chapter 10.)

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 in:
  - ADDSD
  - ALTDSD
  - DELDSD
  - LISTDSD
  - PERMIT
  - SEARCH
  - ICHUT100

If needed, you can use the exits for additional processing of data set names.

The RACF initialization routine, ICHSEC00, locates the ICHNCV00 module and stores the address in the RCVT. If ICHSEC00 finds the module, ICHNCV00 is listed in message ICH508I with the other exit routines.

If you want to change ICHNCV00, you must reassemble it, linkedit it again, and re-IPL.

Command Exits

The command exits allow the installation to associate additional security checking or processing with certain RACF commands or to bypass all security checking. There are two command exits:

- One command exit is called by the RACF commands ADDSD, ALTDSD, DELDSD, LISTDSD, PERMIT, and SEARCH, and by the RACF utility ICHUT100. The exit must be named ICHCNX00. It is entered after syntax checking and, for the:

  - 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, 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, and after each data set profile is located but before any authorization checking is performed.

  - ICHUT100 utility, after the data set profile is retrieved, but before the data set profile is associated with a user or group.

  The ALTDSD, DELDSD, LISTDSD, PERMIT, and SEARCH commands issue RACHECK macros to check the command user's authority to a specified resource. Thus, the RACHECK preprocessing and postprocessing exits will 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.

  RACF processes the data set naming convention table (if any) immediately before the call to the ICHCNX00 exit.

- The other command exit is called by the RACF commands DELUSER, DELGROUP, and REMOVE. The exit must be named ICHCCX00. It is entered after syntax checking and before any data set profile is located.

RACF Report Writer Exit

ICHRSMFE is an optional, installation-written exit routine that you can use to:

- Create additional selection/rejection criteria for records that the RACF report writer processes

- Modify data set naming conventions in records that the RACF report writer processes

- Create additional output reports, in addition to the reports that the RACF report writer provides

To avoid an unresolved external reference from the linkage editor, ICHRSMFE is shipped as a dummy module (BR 14) that is linkedited into the RACF report writer load module, RACFRW.

Each time the RACF report writer reads a record from the SMF data set, 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 the *Auditor's Guide*.

MVS Router Exit

The system authorization facility (SAF) and the MVS router are present on all MVS/SP systems, even if RACF is not installed. Although the MVS router is not part of RACF, many system components and programs will invoke RACF through the RACROUTE macro and SAF. Therefore, installations have the opportunity to modify RACF parameter lists and do customized security processing within the MVS router.

Depending on the level of JES programming support available on your system, JES might invoke the system authorization facility (SAF) after it reads a job. JES issues the RACROUTE macro with the REQUEST=VERIFY and ENVIR=VERIFY parameters. This invocation of SAF does not result in a call to RACF; it provides an opportunity for the installation to install its own userid verification with the MVS router exit.

Summary of Installation Exit Callers

Figure 7-1 lists the macros, commands, and utilities that give control to each exit routine.

Notes:

1. *RALTER and RDEFINE give control to the RACLIST and FRACHECK exits only if the user is not SPECIAL and has specified ADDMEM or DELMEM.*

2. *RACF calls the naming convention exit before it calls the RACDEF preprocessing exit, the RACHECK preprocessing exit, and the command preprocessing exit ICHCNX00.*

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 if the entry address does not correspond with the address specified in the RCVT, DSMON prints an error message.

You can use the information in this report to verify that only those exit routines that have been defined by your installation are active. The existence of any other exit routines might indicate a system security exposure, because RACF exit routines could be used to bypass RACF security checking. Similarly, if the length of an installation-defined exit routine module differs from the length of the module when it was defined by your installation, you should notify your RACF security administrator (because the module might have unauthorized modifications).

Exits Macros, Commands and Utilities	RACDEF pre and post ICHRDX01, 02	RACHECK pre and post ICHRCX01, 02	RACINIT pre and post ICHRIX01, 02	RACLIST pre/post and Selection ICHRLX01, 02	FRACHECK pre and post ICHRFX01, 02	New Password ICHPWX01	Command ICHCNX00	Command ICHCCX00
ADDGROUP								
ADDSD	X						X	
ADUSER								
ALTDSD		X					X	
ALTGROUP								
ALTUSER						X		
CONNECT								
DELDSD	X	X					X	
DELGROUP								X
DELUSER								X
LISTDSD		X					X	
LISTGRP								
LISTUSER								
PASSWORD						X		
PERMIT		X					X	
RALTER		X		X	X			
RDEFINE				X	X			
RDELETE		X						
REMOVE								X
RLIST		X						
RVARY								
SEARCH		X					X	
SETROPTS								
RACDEF	X							
RACHECK		X						
RACINIT			X			X		
RACLIST				X				
FRACHECK					X			
ICHUT100							X	

Figure 7-1. RACF Installation Exits Cross-Reference Table

Possible Uses of RACF Exits

Some possible uses for RACF exits are:

- Allowing access to a RACF-protected resource when RACF is inactive
- Protecting the user from accidental destruction of data
- Enforcing rules for the content of passwords
- Modifying data set naming conventions
- Ensuring tape protection for tapes without a profile

Allowing Access When RACF is Inactive

When RACF is inactive, any attempt to access a protected data set is passed to the RACHECK preprocessing exit. On systems that have always call, this check occurs for all data sets. The exit can determine whether or not to allow the access to proceed. If, for example, you want to allow one or more specific users to perform recovery operations, the exit must select them. (If RACF is inactive, the normal privileges of OPERATIONS cannot be used because the RACF data set may not be available to verify that a user is so authorized.) You should consider whether userids or batch jobnames authorized by the exits when RACF is inactive should also be allowed to use the system if RACF is running normally.

If the RACHECK preprocessing exit routine neither grants nor denies access to the data set, RACF failsoft processing may prompt the operator.

Protecting the User's Resources from the User

Users can accidentally delete their own data. Suppose, for example, that a user wishes to delete a library member, but forgets to include the member name in the command. The user issues:

```
DELETE  USER.LIB
```

instead of

```
DELETE  USER.LIB(progname)
```

With ALTER authority to USER.LIB, the result is the loss of the entire library.

To delete a member requires only UPDATE authority, whereas deletion of the whole library requires ALTER. By default, the creator of a data set (a library is only a special use of a data set) automatically has ALTER authority to it. Therefore, the user is susceptible to this exposure.

For group data sets (libraries), the creator is in the access list explicitly and can therefore take positive steps to reduce his own authority to UPDATE. The creator is then unable to delete the group data set accidentally. When the creator really does want to delete it, then, still being the owner of the data set, he can restore his ALTER authority using the PERMIT command.

While this is very simple for group data sets, user data sets do not have their creators in the access list. The creator has ALTER authority by virtue of the naming convention (the high-level qualifier and userid match). There is, therefore, no userid for the creator to remove, and his data is still vulnerable to his own errors.

To provide the same facility for user data sets, you must use the RACF exits. In the RACHECK exit, you can determine if the user is seeking to scratch one of his own data sets. Under these circumstances the exit can invalidate access with the naming convention (by blanking out the QUALIFIER field and allowing access only as specified in the access list). When a user really wants to delete the entire data set, the user can authorize himself explicitly because he is the owner.

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 validate 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 might be forced) to change passwords. The installation can devise whatever tests it wishes in order 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 might ensure that the password is more than six characters or that it contains an alphanumeric mix. With an exit, more complex tests might disallow names, months, userids, and group names, or detect trivial usage of alphanumeric mixes such as JAN85 and FEB85.

Modifying Data Set Naming Conventions

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 (the high-level qualifier is a userid or group name)
- Conversion of the internal RACF format back to the user's real data set name for display purposes (ICHUT100, and LISTDSD and SEARCH after a profile is located but before it is displayed)
- Identification of a userid or group name to be used for authority checking
- Optionally, enforce other restrictions on data set names (format and content) on define requests (ADDSD, RACDEF DEFINE/RENAME)

- When running with more than one input data set, the ICHUT400 utility copies from the lowest number INDDn data set

Some of the exits that you could use for modifying data set naming conventions are:

- ICHCNX00 - Command preprocessing exit
- ICHCCX00 - Command preprocessing exit
- ICHRDX01 - RACDEF preprocessing exit
- ICHRDX02 - RACDEF postprocessing exit
- ICHRLX01 - RACLIST pre/postprocessing exit
- ICHRLX02 - RACLIST selection exit
- ICHRFX01 - FRACHECK preprocessing exit
- ICHRFX02 - FRACHECK postprocessing exit

RACDEF Preprocessing (ICHRDX01) and Postprocessing (ICHRDX02) Exits

Many of the values passed to the RACDEF preprocessing and postprocessing exits are derived from the parameters specified on the RACDEF macro instruction. For details on this macro instruction, see *SPL: Supervisor* or *SPL: System Macros and Facilities.*

On entry to the RACDEF preprocessing (ICHRDX01) and postprocessing (ICHRDX02) exit routines, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Length address:** points to a fullword containing the number of fullwords in this parameter list.
4	4	**Flag byte address:** points to a 1-byte area of the following format: 00.. .... TYPE=DEFINE was specified (or assumed) 01.. .... TYPE=ADDVOL was specified 10.. .... TYPE=DELETE was specified 11.. .... TYPE=CHGVOL was specified ..1. .... OLDVOL=old vol addr was specified ...1 .... NEWNAME=new dsn addr was specified .... 0... Reserved .... .1.. DSTYPE=V was specified .... ..1. DSTYPE=M was specified ....1 SPECIAL=YES was specified Multiple flags can be set, for example: TYPE=DEFINE and NEWNAME indicates a rename request. TYPE=ADDVOL and OLDVOL indicates a DASD data set is being extended to a new volume.
8	4	**INSTLN address:** points to an area containing the installation parameters. This address is zero if INSTLN was not specified. The INSTLN parameter is not specified by any system modules but is intended for use by installation-written routines that invoke RACDEF to communicate with the RACDEF preprocessing exit routine.
12	4	**ENTITY address:** points to an area containing the resource name. For the DATASET class, this area is 44 bytes long. For general resource classes, the length comes from the class descriptor table. The name is left-justified and padded on the right with blanks. (See note.)
16	4	**OLDVOL or NEWNAME address:** points to a 44-byte area containing the old volume serial number (for OLDVOL) or the new data set name (for NEWNAME). This address points to an area containing blanks if the class is not DATASET. (See note.)
20	4	**VOLSER address:** points to a 6-byte area containing the volume serial number when specified for both the DATASET and TAPEVOL classes. Otherwise, this address points to an area containing blanks. (See note.)
24	4	**CLASS address:** points to an area containing a 1-byte field containing the classname length followed by the entity class name. (See note.)
28	4	**MENTITY address:** points to a 44-byte area containing the name of the profile to be modeled. This area contains blanks if MENTITY was not specified. (See note.) If supplied by the exit, the MENTITY value must be the name of a DATASET profile. The class of the profile being created can be DATASET or any class defined by a class descriptor. If a DATASET profile is found, the following fields are copied from it to the new profile: access list, level, universal access, owner, installation-defined data, and logging options (auditing flags). This processing occurs only for a DEFINE request without NEWNAME. RACDEF's search for the MENTITY profile starts with a chain of resident profiles pointed to from the ACEEAMP field. Profiles are added to this chain by RACDEF depending on the options set in the flag byte pointed to from offset 36 in the parameter list.
32	4	**MVOLSER address:** points to a 6-byte area containing the volume serial number of the data set profile being modeled. This area contains blanks if MVOLSER was not specified or if the class is not DATASET.

36 4 **Flag byte address:** points to a 1-byte area of the following format:

1... .... Continue processing. Treat TYPE=DEFINE and MENTITY not defined to RACF as if MENTITY was not specified.

.1.. .... Add the MENTITY profile to the chain of profiles pointed to by the ACEEAMP field if the profile is found in the RACF data set.

..1. .... Add the MENTITY profile to the chain of profiles pointed to by the ACEEAMP field whether or not the profile is found in the RACF data set. If not found, build a dummy profile with the MENTITY name, MVOLSER value, and not-found indicator.

...1 .... Add the MENTITY profile to the chain of profiles pointed to by the ACEEAMP field if the profile is not found in the RACF data set. Build a dummy profile with the MENTITY name, MVOLSER value, and not-found indicator.

.... 0000 Reserved.

40 4 **Naming conventions address:** points to the parameter list of the ICHCNX00 exit. The ICHCNX00 exit, invoked by RACF commands and the ICHUT100 utility, allows an installation to modify or eliminate the RACF DASD data set naming convention. Corresponding processing might be required in the RACDEF preprocessing and postprocessing exits, so a parameter list with similar structure and content is passed to them to allow use of common routines.

44 4 **Profile options flag byte address:** points to a 1-byte area of the following format:

1... .... Use the UACC value from the installation-supplied profile.

.1.. .... Use the LEVEL value from the installation-supplied profile.

..1. .... Use the OWNER value from the installation-supplied profile.

...1 .... Use the AUDIT value from the installation-supplied profile.

.... 1... Use the GLOBALAUDIT value from the installation-supplied profile.

.... .1.. Use the installation data from the installation-supplied profile.

.... ..1. Use the access list from the installation-supplied profile.

....1 Use the WARNING value from the installation-supplied profile.

48 4 **Installation-supplied profile address:** points to a profile in the format of that returned by RACHECK ENTITY=(addr,CSA). The profile options flag byte determines the values used in this profile. (On entry to the preprocessing exit, this address is zero.) If both MENTITY processing and installation-supplied profile processing are requested, values from the MENTITY profile override RACDEF defaults and values specified for use from the installation-supplied profile override MENTITY profile values.

52 4 **ACEE address:** points to a fullword containing the address of an ACEE that will be used for RACDEF processing. If the ACEE parameter was not specified on the RACDEF macro instruction or changed by the exit, the fullword pointed to by this value contains zeroes and RACDEF processing uses the ACEE pointed to by TCBSENV in the current task control block (TCB) or ASXBSENV in the address space extension block (ASXB).

56 4 **UNIT Information address:** points to an area prefixed by a one-byte length field that contains the length of the UNIT information. If the length is 4, it is assumed the UNIT information contains the UCB coded information. If the length is 5 to 8 characters, it is assumed the UNIT information contains the generic unit information (such as 3330-1 or SYSDA). If this address value is zero or the length field is zero, it is assumed that UNIT information is absent. (See note.)

60 4 **UACC address:** points to a one-byte area containing the universal access authority to be placed in the resource profile being defined. (See note.) The UACC value has the following format:

1... .... ALTER authority

.1.. .... CONTROL authority

..1. .... UPDATE authority

...1 .... READ authority

.... 000. Reserved

....1 NONE authority

64 4 **LEVEL address:** points to a one-byte area containing the level value to be placed in the new resource profile. This value must be in the range of 00 to 99.

68 4 **AUDIT address:** points to a one-byte area containing the audit flags to be placed in the new resource profile. The AUDIT flag area has the following format:

1...	....	Audit all accesses.
.1..	....	Audit all successful accesses.
..1.	....	Audit all access attempts that fail.
...1	....	No auditing.
....	xx..	Qualifier for successful access attempts.
....	..xx	Qualifier for unsuccessful access attempt.

The qualifier is of the following format:

Value	**Access Level**
00	READ
01	UPDATE
10	CONTROL
11	ALTER

72 4 **OWNER address:** points to an eight-byte area containing the owner name to be placed in the new resource profile. This owner name must be a RACF-defined userid or group name. If there is no owner, this field contains blanks or zeroes to indicate the information is absent.

76 4 **DATA address:** points to a variable length area of the following format:

Offset 0, length 1: Length of data information
Offset 1, variable length: data information

If no data information was supplied, the length is zero and the value is blanks, so that an exit routine can supply the information.

80 4 **Flag Byte 2 Address:** points to a 1-byte area of the following format. (See note.)

00..	....	RACFIND was not specified
10..	....	RACFIND = NO was specified
11..	....	RACFIND = YES was specified
..1.	....	CHKAUTH = YES was specified
...1	....	DSTYPE = TAPE was specified
....	1...	ERASE = YES was specified
....	.0..	MGENER = ASIS was specified
....	.1..	MGENER = YES was specified
....	..1.	WARNING = YES was specified
....	...1	GENERIC = YES was specified

84 4 **completion code address:** points to a 4-byte field containing the ABEND code that RACDEF is going to issue. The completion code is contained in the low-order 12 bits of the field. The address points to an area containing zeroes if RACDEF is not going to issue an ABEND. (If ABEND processing is to be bypassed by RACDEF, the exit routine can zero the completion code. In this case, the exit routine should also set the return code to zero; otherwise, the ABEND reason code will be passed to the RACDEF caller as a return code.) Do not confuse an ABEND issued by RACDEF with one issued by an invoker of RACDEF. If a user is not authorized to a resource, RACDEF will not issue an ABEND, but the invoker of RACDEF might. For example, OPEN might issue a 913 ABEND in this case, although RACDEF completed without any ABEND.

88 4 **Return code address:**: points to a 4-byte field containing either the return code to be passed back to the RACDEF caller in response to the define request (for the meanings of these return codes, see *SPL: Supervisor* or *SPL: System Macros and Facilities*) or the reason code used to cause the ABEND to be issued (for the meanings of these ABEND reason codes, see *RACF Messages and Codes*). Do not confuse this code with the return code from the RACDEF preprocessing or postprocessing exit routines described in this chapter.

92 4 **Return code address:**: points to a 4-byte field containing the return code to be passed back to the RACDEF caller in response to the define request (for the meanings of these return codes, see *SPL: Supervisor* or *SPL: System Macros and Facilities*).

96 4 **exit work area**: points to a fullword of zeroes on the initial entry to the preprocessing routine. An installation can use this field for any purpose. Because this field is set to zeroes before entry to the preprocessing exit, the preprocessing and postprocessing exits can use this work area to communicate with each other.

100	4	**third flag byte**: points to a 1-byte area of the following format. 00.. .... TAPELBL = STD was specified 01.. .... TAPELBL = NL was specified 10.. .... TAPELBL = BLP was specified .... ..0. If retention period is present, halfword number of days for retention period. ... ..1. If retention period is present, it is formatted as a 3 byte expiration date.
104	4	**address of ACCLVL value**: points to a 1-byte length field followed by 0 to 8 bytes of data specified by the first subparameter of the ACCLVL parameter on the RACDEF or RACROUTE REQUEST = DEFINE macro
108	4	**address of ACCLVL parameter**: points to a 1-byte length field followed by 0 to 8 bytes of data specified by the second subparameter of the ACCLVL parameter on the RACDEF or RACROUTE REQUEST = DEFINE macro
112	4	**address of SECLVL parameter** : points to a fullword count field followed by the same number of security level (SECLVL) values (currently either 0 or 1). This security level is the same as the numeric part of the installation-defined security level
116	4	**address of CATEGORY parameter**: points to a fullword count field followed by the same number of binary halfword category values. Each category value identifies an installation-defined value.
120	4	**address of file sequence number**: points to a 2-byte field containing the file sequence number for a tape data set
124	4	**address of security retention period**: points to a 2 or 3-byte field containing the retention period or expiration date depending on the bit settings in the third byte flag at offset 100.
128	4	**address of NOTIFY userid**: points to an 8-byte area containing the userid of the user to be notified when RACF detects an unauthorized attempt to access a resource protected by this profile

Note: If the exit changes this value, the RACF profile being processed is changed, but RACF does not communicate the change to the invoker of RACDEF. For example, if a data set is being defined to RACF and the exit changes the entity value, the RACF profile defined will show the data set itself is unchanged. Similar processing applies to the OLDVOL, NEWNAME, VOLSER, UNIT, UACC, LEVEL, AUDIT, OWNER, DATA, and CLASS parameters.

Return Codes - 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 RACDEF SVC (the meanings of the RACDEF return codes are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities.*)

Hex (Decimal)	Meaning
0 (0)	Exit routine processing is complete. Normal SVC processing is to continue.
4 (4)	The request is not accepted and is to be failed.
8 (8)	The request is accepted. No more SVC processing is to be performed.
C (12)	The request is accepted. SVC processing continues, but authorization checking is bypassed.

Notes:

1. *If register 15 contains any other value, RACDEF issues an completion code (385) that indicates an invalid exit return code.*

2. *Return codes 0, 8, and C cause the RACDEF SVC to issue a return code of 0 to the module calling the SVC.*

Return Codes - 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 RACDEF SVC (the meanings of the RACDEF return codes are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities.*)

Code	Meaning
0	Exit routine processing is complete. Normal SVC processing is to continue.
4	Retry the RACDEF SVC; invoke the RACDEF preprocessing routine. (The return code, reason code, completion code, access authority, owner, level, and auditing fields are reset to zeroes before a retry.)

Note: If register 15 contains any other value, RACDEF issues an completion code (385) that indicates an invalid exit return code.

RACHECK Preprocessing (ICHRCX01) and Postprocessing (ICHRCX02) Exits

Many of the values passed to the RACHECK preprocessing and postprocessing exit are derived from the parameters specified on the RACHECK macro instruction. For details on the RACHECK macro instruction, see *SPL: Supervisor* or *SPL: System Macros and Facilities.*

On entry to the RACHECK preprocessing (ICHRCX01) and postprocessing (ICHRCX02) exit routines, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Length address:** points to a fullword containing the number of fullwords in this parameter list.
4	4	**Flag byte 1 address:** points to a 1-byte area of the following format:
		00.. .... RACFIND was not specified.
		10.. .... RACFIND=NO was specified.
		11.. .... RACFIND=YES was specified.
		..0. .... Reserved.
		...1 .... DSTYPE=V was specified.
		.... 0... Reserved.
		.... .1.. LOG=NOFAIL was specified.
		.... ..1. LOG=NONE was specified.
		....1 ENTITY=(entity name addr, CSA) was specified.
8	4	**Flag byte 2 address:** points to a 1-byte area of the following format:
		1000 0000 ATTR=ALTER was specified.
		0000 1000 ATTR=CONTROL was specified.
		0000 0100 ATTR=UPDATE was specified.
		0000 0010 ATTR=READ was specified (or assumed).
		This value is derived from the ATTR parameter on the RACHECK macro instruction. Note that bit mapping for ATTR differs from bit mapping for the access code (pointed to from offset 48 in the parameter list), which matches the mapping in the RACF data set.
12	4	**Flag byte 3 address:** points to a 1-byte area of the following format:
		0... .... DSTYPE=T
		.1.. .... DSTYPE=M was specified.
		..0. .... ENTITY=dsname; tape volser or DASD volser addr was specified.
		..1. .... PROFILE=profile addr was specified.
		...0 0... Reserved.
		.... .1.. GENERIC=YES was specified.
		.... ..1. Private area profile requested.
		....0 Reserved.
16	4	**INSTLN address:** points to an area containing the installation parameters. This address is zero if INSTLN was not specified. None of the system modules specify the INSTLN parameter. It is intended for use by installation-written routines that invoke RACHECK to communicate with the RACHECK preprocessing and postprocessing exit routines. Do not confuse this value with the DATA address (pointed to from offset 32 in the parameter list) that comes from a field in the RACF profile for the resource being checked.
20	4	**ENTITY or PROFILE address:** points to an area containing the resource name (for ENTITY) or an area containing the profile (for PROFILE). If ENTITY is used, this area is 44 bytes long for the DATASET class. For general resource classes, the length is taken from the class descriptor table. The name or number is left-justified and padded on the right with blanks. If the exit changes this value, the RACF profile affected is changed but RACF does not communicate the change to the invoker of RACHECK. For example, if a user's authority to a data set is being checked and the exit changes the entity value, the RACF profile checked is the one named by the changed value, but the data set itself is unchanged. Similar processing applies to the OLDVOL, VOLSER, OWNER, and CLASS parameters. Note: If you change the entity name, also change the qualifier, whose address is at offset 32 in the ICHCNX00 parameter list, to reflect this change.

24	4	**CLASS address:** points to an area containing a 1-byte length field containing the classname length followed by a field containing the entity class name.
28	4	**VOLSER address:** points to a 6-byte area containing the volume serial number. This address points to an area containing blanks if the class is not DATASET.
32	4	**DATA address:** points to a 1-byte length field followed by the installation data for the entity specified on RACHECK. This address is zero for the preprocessing routine. This address is zero for the postprocessing routine if (1) no data is present, (2) the profile could not be retrieved, or (3) the preprocessing routine indicated bypassing of RACHECK.
36	4	**Work area address:** points to a fullword of zeroes on the initial entry to the preprocessing routine. An installation can use this field for any purpose. Because this field is set to zeroes before entry to the preprocessing exit, the preprocessing and postprocessing exits can use this work area to communicate with each other.
40	4	**ABEND code address:** points to a 4-byte field containing the ABEND code that RACHECK is going to issue. The ABEND code is contained in the low-order 12 bits of the field. The address points to an area containing zeroes if RACHECK is not going to issue an ABEND. (If ABEND processing is to be bypassed by RACHECK, the exit routine can zero the ABEND code. In this case, the exit routine should also set the return code to zero; otherwise, the ABEND reason code will be passed to the RACHECK caller as a return code.) Do not confuse an ABEND issued by RACHECK with one issued by an invoker of RACHECK. If a user is not authorized to a resource, RACHECK will not issue an ABEND, but the invoker of RACHECK might. For example, OPEN might issue a 913 ABEND in this case, although RACHECK completed without any ABEND.
44	4	**Return code address:** points to a 4-byte field containing either the return code to be passed back to the RACHECK caller in response to the access request (for the meanings of these return codes, see *SPL: Supervisor* or *SPL: System Macros and Facilities*) or the reason code used to cause the ABEND to be issued (for the meanings of these ABEND reason codes, see *RACF Messages and Codes*). Do not confuse this code with the return code from the RACHECK preprocessing or postprocessing exit routines described in this chapter.
48	4	**Access code address:** points to a 1-byte field containing the user's authorization to the resource that is being checked: X'80' - ALTER X'40' - CONTROL X'20' - UPDATE X'10' - READ X'01' - NONE The area is zero if (1) the profile could not be retrieved, or (2) the preprocessing routine indicated bypassing of RACHECK.
52	4	**Resource level number address:** points to a 1-byte field containing the LEVEL value from the resource profile. This address is zero for the preprocessing routine. This address is zero for the postprocessing routine if (1) the profile could not be retrieved, or (2) the preprocessing routine indicated bypassing of RACHECK.
56	4	**OLDVOL address:** points to a 6-byte area containing the volume serial number of a previously defined volume of a multivolume data set or tape volume set. This is blank if OLDVOL was not specified.
60	4	**Naming conventions address:** points to the parameter list of the ICHCNX00 exit. The ICHCNX00 exit invoked by RACF commands and the ICHUT100 utility allows an installation to modify or eliminate the RACF DASD data set naming convention. Corresponding processing might be required in the RACHECK preprocessing exit, so a parameter list with similar structure and content is passed to it to allow the use of common routines.
64	4	**APPL name address:** points to an eight-byte field containing the application name (if supplied on the RACHECK macro instruction). The name is left-justified and padded with blanks. If the APPL parameter was not specified, the field contains blanks. RACHECK processing does not reference this field; this field is intended to provide additional information for the exit routines.
68	4	**ACEE address:** points to a fullword containing the address of the ACEE that is used for RACHECK processing. If the ACEE parameter was not specified on the RACHECK macro instruction, the fullword pointed to by this value contains zeroes, and the ACEE pointed to by TCBSENV in the current task control block (TCB) or ASXBSENV in the address space extension block (ASXB) is used for authority checking.

72	4	**OWNER address:** points to an eight-byte area containing an identifier that is to be compared with the OWNER field in the resource profile whose access is being checked. If the OWNER parameter was not specified on the RACHECK macro instruction, the area pointed to by this address contains blanks. Note that use of the owner field causes RACHECK to bypass checking of the OPERATIONS attribute during authority checking.
76	4	**Logging control address:** points to a fullword that the postprocessing exit can use to control auditing. On entry, the fullword is set to zero. The exit may change this value to 4 to unconditionally request logging or to 8 to unconditionally suppress logging. (Note that you can never override the GLOBALAUDIT option.)
80	4	**ACCLVL value address:** points to a 1-byte length field followed by 0 to 8 bytes of data from the first subparameter in the ACCLVL keyword on the RACHECK macro.
84	4	**ACCLVL parameter list address:** points to the parameter list passed as the second subparameter in the ACCLVL keyword on the RACHECK macro.
88	4	**address of file sequence number** points to a two-byte field containing the file sequence number for a tape data set.
92	4	**address of tape flag byte**: points to a 1-byte area of the following format: 10.. .... TAPELBL=BLP was specified. 01.. .... TAPELBL=NL was specified. 00.. .... TAPELBL=SL was specified. ..00 0000 Reserved.
96	4	**address of fourth flag byte**: points to a 1-byte area of the following format: 1... .... STATUS=ERASE was specified. .000 0000 Reserved.
100	4	**RACHECK reason code address**: points to a 4-byte field containing the reason code to be used with the return code pointed to by offset 44. See *SPL System Macros and Facilities* for the meanings of the RACHECK reason codes. Do not confuse this reason code with the ABEND reason code.
104	4	**address of NOTIFY userid** an 8-byte area containing the userid of the user to be notified when RACF detects an unauthorized attempt to access a resource protected by this profile. This field is valid only for the postprocessing exit and only if the PROFILE specified NOTIFY.

Return Codes - 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 RACHECK SVC, the meanings of which are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities.*

Hex (Decimal)	Meaning
0 (0)	Exit routine processing is complete. Normal SVC 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 SVC 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 (for example, logging) is to continue.

Note: If register 15 contains any other value, RACHECK issues an ABEND code (382) that indicates an invalid exit return code.

RACF uses resident profiles for three purposes:

- As model profiles, as specified by the MENTITY and MVOLSER parameters on RACDEF. These profiles are built by RACDEF when an MENTITY and MVOLSER value are supplied (either by a caller or by an exit routine) and the exit routine requests that the profile be retrieved and added to a chain of profiles pointed to by the ACEEAMP field. RACINIT delete processing releases the storage used by these profiles.

- As installation-supplied profiles, as specified by an exit routine.

The ICHRRPF macro maps the resident profile. See the RRPF mapping in Chapter 12.

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).

For more information on the format, see the RRPF data area in Chapter 12.

Return Codes - RACHECK Postprocessing Exit

When the RACHECK preprocessing exit routine returns a return code of 4 or 8, and when the RACHECK macro instruction request specified ENTITY=(entity address, CSA) or a private area profile was requested (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.

Do not confuse these return codes with the return code from the RACHECK SVC, the meanings of which are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities*

When the RACHECK postprocessing exit routine returns control, register 15 should contain one of the following return codes:

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 an invalid exit return code.

RACINIT Preprocessing (ICHRIX01) and Postprocessing (ICHRIX02) Exits

Note: Many of the values passed to the RACINIT preprocessing and postprocessing exits are derived from the parameters specified on the RACINIT macro instruction. For details on this macro instruction, see *SPL: Supervisor* or *SPL: System Macros and Facilities.*

On entry to the RACINIT preprocessing (ICHRIX01) and postprocessing (ICHRIX02) exit routines, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Length address:** points to a fullword containing the number of fullwords in this parameter list.
4	4	**Flag byte address:** points to a 1-byte area of the following format: 00.. .... ENVIR=CREATE was specified (or assumed). 01.. .... ENVIR=CHANGE was specified. 10.. .... ENVIR=DELETE was specified. ..0. .... SMC=YES was specified. ..1. .... SMC=NO was specified. ...1 .... SUBPOOL parameter specified. .... 0... PASSCHK=YES was specified. .... 1... PASSCHK=NO was specified (bypass password checking). Both the preprocessing and postprocessing exit can set this option. This option causes RACINIT to: — Bypass checking that the old password is correct and has not expired — Bypass checking that the new password is valid — Bypass updating the old password with the new If PASSCHK=NO, the postprocessing exit must issue a return code of 4 to re-invoke the RACINIT function to allow the option to take effect. An installation can use this procedure to bypass enforcing password expiration. .... .1.. STAT=NO was specified .... ..1. LOG=ALL was specified ....1 ENCRYPT=NO was specified.
8	4	**USERID address:** points to an area of the following format: Offset 0, length 1: Length of user identification Offset 1, length 8: User identification If no userid was supplied, the length is zero and the value is blanks, so that an exit routine can supply a value. If a started procedure name was supplied, then the userid might have come from the started procedure table (ICHRIN03). See Chapter 5, "RACF Options."
12	4	**PASSWORD address:** points to an area of the following format: Offset 0, length 1: Length of password Offset 1, length 8: Password If ENCRYPT=NO was specified, the password is treated as if it is already encrypted. If no password was supplied, the length is zero and the value is blanks, so that an exit routine can supply a value.
16	4	**START address:** points to an 8-byte area containing the PROC name of the started task. If no started procedure name was supplied, the value is blanks so that an exit routine can supply a value. However, RACINIT will not use the value.
20	4	**INSTLN address:** points to an area containing the installation parameters. This address is zero if INSTLN was not specified. No system modules specify the INSTLN parameter. INSTLN is intended for use by installation-written routines that invoke RACINIT to communicate with the RACINIT preprocessing exit routine. Do not confuse this value with the terminal data value (pointed to from offset 68 in the parameter list) or the user data value (pointed to from offset 72 in the parameter list), which are taken from fields in the RACF profiles for the user entering the system and the terminal being used.

24	4	**GROUP address:** points to an area of the following format: Offset 0, length 1: Length of group name Offset 1, length 8: Group name If no group name was supplied, the length is zero and the value is blanks, so that an exit routine can supply a value.
28	4	**NEWPASS address:** points to an area of the following format: Offset 0, length 1: length of new password Offset 1, length 8: new password If no new password was supplied, the length is zero and the value is blanks, so that an exit routine can supply a value.
32	4	**ACEE address:** points to an area containing the access control environment element. At entry to the RACINIT preprocessing exit, this address points to the area of storage where the ACEE will be built. At entry to the RACINIT postprocessing exit, this address points to the actual ACEE built by RACINIT. Any changes made by the postprocessing routine remain in effect for the duration of the session or job.
36	4	**PGMNAME address:** points to a 20-byte area containing the programmer name information (or blanks if no programmer name exists). This address is zero if PGMNAME was not specified.
40	4	**ACTINFO address:** points to a 144-byte area containing accounting information (or zeroes if no accounting information exists). The 144-byte area is consistent with similar accounting information in the SMF (type 20) job initiation record: — The first byte contains the number (in binary) of accounting fields. — The following bytes contain accounting fields, where each entry for an accounting field contains a 1-byte length field followed by the field (in EBCDIC). A length indicator of 0 indicates an omitted field. This address is zero if ACTINFO was not supplied.
44	4	**OIDCARD address:** points to an area containing a 1-byte length field followed by a field containing the OIDCARD identification number. The length byte is 0 if OIDCARD was not specified.
48	4	**TERMID address:** points to an 8-byte area containing the terminal identifier. The name is left-justified and padded on the right with blanks. This address is 0 if TERMID was not specified.
52	4	**Work area address:** points to a fullword of zeroes on the initial entry to the preprocessing routine. Because this field is set to zeroes before entry to the preprocessing exit, the preprocessing and postprocessing exits can use this work area to communicate with each other.
56	4	**ABEND code address:** points to a 4-byte field containing the ABEND code that RACINIT is going to issue. The low-order 12 bits of the field contain the ABEND code. The address points to an area containing zeroes for the postprocessing routine if RACINIT is not going to issue an ABEND code. This address points to an area containing zeroes for the preprocessing routine. (If ABEND processing is to be bypassed, the exit routine can set the ABEND code to zero. In this case, the return code should also be set to zero; otherwise, the ABEND reason code will be passed to the RACINIT caller as a return code). Do not confuse an ABEND issued by RACINIT with one issued by an invoker of RACINIT. For example, if a user is not defined to RACF, RACINIT will not issue an ABEND, but the invoker of RACHECK may. A batch job might fail with a JCL error in this case, although RACINIT completed without an ABEND.
60	4	**Return code address:** points to a 4-byte field containing either the return code to be passed back to the RACINIT caller in response to the identification request or the reason code that causes the ABEND to be issued (if nonzero). This address points to an area containing zeroes for the preprocessing routine. Do not confuse this return code with the return code from the RACINIT pre/postprocessing exit routines, the meanings of which are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities.*
64	4	**Flag byte address:** points to a 1-byte area of the following format: 1... .... Bypass OIDCARD processing. RACINIT will ignore any OIDCARD information and any user profile indication that an OIDCARD is required. .000 0000 Reserved.

68	4	**Terminal data address:** points to a 1-byte length field followed by the installation terminal data, as specified in the DATA parameter of the RDEFINE or RALTER commands. The length field includes the 1-byte length of the length field. This address is zero if (1) no data is present, (2) the profile could not be retrieved, (3) the preprocessing routine indicated bypassing of RACINIT, or (4) the NOTERMINAL system option is in effect. 72 4 **User data address:** points to a 1-byte length field followed by the installation data specified on the ADDUSER and ALTUSER commands for the user specified on RACINIT. The length field includes the 1-byte length of the length field. The address points to an area containing zeroes for the preprocessing routine. In addition, the address is zero if (1) no data is present, (2) the profile could not be retrieved, or (3) the preprocessing routine indicated bypassing of RACINIT.
76	4	**Terminal level number address:** points to a 1-byte field containing the LEVEL value from the terminal profile as set by the RDEFINE or RALTER commands. This address is zero if (1) the profile could not be retrieved, or (2) the preprocessing routine indicated bypassing of RACINIT.
80	4	**Jobname address:** points to an 8-byte area containing the job name of a background job. The area contains blanks if no job name information is available.
84	4	**APPL name address:** points to an 8-byte field containing the application name, if supplied on the RACINIT macro instruction. The name is left-justified and padded with blanks. If the APPL parameter was not specified, the field contains blanks.
88	4	**SUBPOOL address:** points to a 1-byte field containing the subpool (as specified on the RACINIT macro) from which the ACEE and its related storage will be obtained. This field has meaning only when the appropriate bit is set in the flag byte, pointed to from offset 4 in the parameter list. Because the storage has already been obtained when the preprocessing exit gains control, there is no effect if the exit changes this value.
92	4	**ACEE address:** points to a fullword containing the address specified on the ACEE parameter of the RACINIT macro instruction. If the ACEE parameter was not specified on the RACINIT macro instruction, this parameter is zero. When specified, the fullword has the following meanings: — For ENVIR=CREATE, RACF will place the address of the ACEE to be built in the fullword and not into the ASXBSENV. This address is identical to the contents of the field at offset 32 in the parameter list. — For ENVIR=CHANGE or ENVIR=DELETE, the fullword contains the address of the ACEE as specified on the RACINIT macro instruction. This address is identical to the contents of the field at offset 32 in the parameter list.
96	4	**Application data pointer:** points to a 1-byte field containing the length of the application data followed by the application data as specified by the DATA operand on the RDEFINE or RALTER commands. The pointer is zero if 1) the application name was not supplied, 2) the profile could not be retrieved, or 3) the preprocessing exit indicated bypassing of RACINIT. The pointer is always zero on entry to the preprocessing exit.
100	4	**Application level pointer:** points to a 1-byte field containing the level value for the application, as specified by the LEVEL operand on the RDEFINE or RALTER commands. The pointer is zero if: — The application name was not supplied — The profile could not be retrieved — The preprocessing exit indicated bypassing of RACINIT The address is always zero on entry to the preprocessing exit. (*Note:* If the application identified at offset 84 is IMS, you should not use this field because IMS uses this field when IMS is active.)
104	4	**Password Change Interval Address:** points to a 4-byte area that contains a 31-bit fixed binary integer that represents the password change interval value found in the user's profile.
108	4	**Password Last Change Date Address:** points to a 3-byte area that contains the date of the last password change. The format of this area is: yyddds where: 'yy' is the year 'ddd' is the day 's' is the packed decimal sign.

Return Codes - 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 SVC processing is to continue.
4	The request is not accepted and is to be failed.
8	The request is accepted. No more SVC processing is performed; however, the postprocessing exit is still invoked.

Note: If register 15 contains any other value, RACINIT issues an ABEND code (383) that indicates an invalid exit return code.

Do not confuse codes from the RACINIT preprocessing exit routine with the return codes from the RACINIT SVC, which are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities.*

When the RACINIT preprocessing exit routine sets a return code of 8, and when the RACINIT macro instruction request specified ENVIR=CREATE, the exit routine is responsible for creating and initializing the access control environment element (ACEE). In addition:

- The 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, then the exit must free the ACEE passed 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.

Return Codes - 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 return or ABEND code values, RACINIT will use the changed values.)
4	Try the RACINIT SVC 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 pre/postprocessing exit routine.)

Note: If register 15 contains any other value, RACINIT issues an ABEND code (383) that indicates an invalid 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 RACINIT SVC without adjusting the input parameters accordingly might cause RACINIT failure.

Do not confuse return codes from the RACINIT postprocessing exit routine with return codes from the RACINIT SVC, which are documented in *SPL: Supervisor* and *SPL: System Macros and Facilities.*

RACLIST Pre/Postprocessing Exit (ICHRLX01)

On entry to the RACLIST pre/postprocessing exit routine (ICHRLX01), register 1 contains the address of the following parameter list:

Offset	Length	Description
0	4	**Number address:** points to a fullword containing the number of parameters in this list, including itself.
4	4	**Flag byte address:** points to a 1-byte area with the following format:
		00.. .... ENVIR=CREATE
		10.. .... ENVIR=DELETE
		..0. .... OWNER=NO
		..1. .... OWNER=YES
		...0 0000 Reserved
8	4	**Function byte address:** points to a 1-byte area with the following format:
		0... .... Call is for preprocessing
		1... .... Call is for postprocessing
		.000 0000 Reserved
12	4	**INSTLN address:** points to an area containing the data specified by the INSTLN parameter on the RACLIST macro. This address is 0 if INSTLN was not specified on RACLIST. No system modules specify the INSTLN parameter; it is intended for use by installation-written routines that invoke RACLIST to communicate with the RACLIST pre/postprocessing exit routine.
16	4	**CLASS address:** points to an 8-byte field containing the class name. The class name is left-justified and padded with blanks if necessary.
20	4	**ACEE address:** points to a fullword that contains the address of the ACEE as specified on the RACLIST macro. If the ACEE parameter was not specified on the RACLIST macro, the fullword contains zeroes and RACLIST uses the ACEE pointed to by TCBSENV in the current task control block (TCB) or ASXBSENV in the address space extension block (ASXB).
24	4	**APPL address:** points to an 8-byte area containing the application name as specified on the RACLIST macro. If not specified on the RACLIST macro, the 8-byte area contains blanks.
28	4	**SUBPOOL address:** points to a 2-byte area containing subpool information. The first byte identifies the subpool from which the in-storage profile index will be obtained. The second byte identifies the subpool from which the profiles will be obtained. The subpool values are taken from the SUBPOOL parameter on the RACLIST macro instruction. These values can be changed if the exit has been invoked for preprocessing; if the exit is invoked for postprocessing, changes will have no effect, because the storage has already been obtained.
32	4	**LIST address:** specifies the address of a fullword containing the address of the list of resource names specified on the RACLIST macro. The fullword contains zeroes if LIST was not specified on the RACLIST macro. The first halfword of the list of resource names contains the number of resource names in the list. This count field is followed by the resource name entries. Each resource name entry consists of a 1-byte length field giving the length of the resource name followed by the resource name itself.
36	4	**Rule flags address:** points to nine contiguous 1-byte fields. RACLIST processing builds a working profile for each resource name and, for successive occurrences of the resource name, merges the new information with the information in the working profile. A resource name can appear in more than one resource group and can also have a profile of its own on the RACF data set. These rule flags determine how conflicts are resolved between multiple occurrences of resource names within resource groups or between resource groups and a resource profile.

The fields in each of the nine flag bytes have the following significance:

1000	....	Least restrictive when resolving conflicts between occurrences in groups. (For example, a profile with UACC=UPDATE would be selected over one with UACC=NONE.)
0100	....	Most restrictive when resolving conflicts between occurrences in groups. (In this case, UACC=NONE would be selected over UACC=UPDATE.)
0010	....	Use value from external profile when resolving conflicts between occurrences in groups.
0001	....	Use value from working profile when resolving conflicts between occurrences in groups. This rule means that the first value encountered is used. Note that for multiple occurrences of a resource name, the order in which they appear is dependent on the alphanumeric sequence of the resource group names and the individual profile name (if any).
....	1000	Least restrictive when resolving conflicts between groups and an individual occurrence.
....	0100	Most restrictive when resolving conflicts between groups and an individual occurrence.
....	0010	Use value from external profile when resolving conflicts between groups and an individual occurrence.
....	0001	Use value from working profile when resolving conflicts between groups and an individual occurrence. This rule means that the first value encountered is used. Note that, for multiple occurrences of a resource name, the order in which they appear is dependent on the alphanumeric sequence of the resource group names and the individual profile name (if any).

The nine flag bytes are initialized as follows:

UACC: initialized to X'44', meaning use the most restrictive of the profile UACC authorizations.

AUDIT flags: initialized to X'44', meaning OR the flag bytes. This causes an audit option to be in effect in the final profile if it was in effect in *any* of the profiles being merged. RACLIST uses the most-encompassing audit qualifiers. If changed to X'88' by the exit, it would mean AND the flag bytes. This causes an audit option to be in effect in the final profile only if it was on in *all* of the profiles being merged. RACLIST uses the least-encompassing audit qualifiers.

GLOBALAUDIT flags: initialized to X'44' with meaning and effects identical to AUDIT flags.

Resource level: initialized to X'44' meaning use the higher level. If changed to X'88' by the exit, it would mean use the lower level. See the description of the LEVEL operand in the *Command Language Reference.*

Installation data: initialized to X'22', meaning use the value from the external profile. (Bit settings 1000 .... and 0100 .... are treated as 0001 ....; bit settings ....1000 and ....0100 are treated as .... 0001.)

Application data: initialized to X'22', meaning use the value from the external profile.
(Bit settings 1000 .... and 0100 .... are treated as 0001 ....; bit settings ....1000 and .... 0100 are treated as .... 0001.)

Access list entries: initialized to X'88', meaning use the least restrictive of the entries.

OWNER: initialized to X'22', meaning use the value from the external profile. (Bit settings 1000 .... and 0100 .... are treated as 0001 ....; bit settings ....1000 and ....0100 are treated as .... 0001.)

NOTIFY: initialized to X'22', meaning use the value from the external profile. (Bit settings 1000 .... and 0100 .... are treated as 0001 ....; bit settings ....1000 and ....0100 are treated as .... 0001.)

If a RACF selection exit is active, RACLIST uses the value from the working profile for all fields except the access list entries. For the other values, processing proceeds as if X'11' had been specified. The RACLIST selection can override this processing. See the "RACLIST Selection Exit" later in this chapter for more information.

40 4 **Profile expansion amount address:** points to a fullword initialized to zeroes, which is the minimum amount of expansion space to provide at the end of the working profile passed to the processing exit. This parameter allows an exit routine to control how much data it can store in the working profiles that RACLIST builds and passes to the selection exit.

44	4	**Return code address:** specifies the address of a fullword to be used as a return code by RACLIST if the exit issues return code 4. On entry to the preprocessing exit, the return code is 0. On entry to the postprocessing exit, the return code is the value RACLIST would return to the caller. Note that this field allows the exit to terminate RACLIST with a 0 or any other return code. Because the exit routine can build resident profiles and an index structure of its own, it might terminate RACLIST in a non-error case with a normal return code.
48	4	**Work area address:** points to a fullword of zeroes. The exit can use this field for any purpose. Because this field is initialized to zeroes before entry to the pre/postprocessing exit, it can be used for communication between invocations of the exit for preprocessing and postprocessing.
52	4	**Class tree anchor element address:** points to a fullword that contains the address of the class tree anchor element that is added to the class tree anchor element chain pointed to by the effective ACEE (see "ACEE address" at offset 20). Each class chained off the ACEE has one class tree anchor element, containing the classname for the in-storage profiles and a pointer to the in-storage profile structure. This field has meaning only for the RACLIST postprocessing exit and is 0 if a class tree is not encountered. See Chapter 12 for the description of the RACF in-storage profile (ISP).

Return Codes - RACLIST Pre/Postprocessing Exit

On return from the RACLIST pre/postprocessing 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 used 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 as code 4.

Any other return code is treated as an error and RACLIST returns to its caller with a return code of 14 (hex).

RACLIST Selection Exit (ICHRLX02)

On entry to the RACLIST selection exit (ICHRLX02), register 1 contains the address of the following standard parameter list. With the exception of the working profile and the fields at offsets 40 and 48 in the preprocessing exit parameter list, you should not modify fields addressed by this parameter list.

Offset	Length	Description
0	4	**Length address:** points to a fullword containing the number of parameters (including itself) in this list.
4	4	**Preparameters address:** address of the parameter list passed to the RACLIST preprocessing exit routine. This parameter is passed to allow communication between the RACLIST pre/postprocessing exit routines and the RACLIST selection exit routine, because processing logic in one exit routine might require corresponding processing in the other exit routine.
8	4	**Flag byte address:** points to a flag byte with the following format: 0... .... The resource was not previously encountered 1... .... The resource was previously encountered .0.. .... Data comes from a group profile .1.. .... Data comes from a resource profile ..00 0000 Reserved
12	4	**Resource name address:** points to a 1-byte length field followed by the name of the resource that RACLIST is currently processing.
16	4	**Resource group name address:** points to a 1-byte length field followed by the name of the resource group from which the current resource name was selected. This address is 0 if the resource name is not from a resource group. The exit should not change this value. Do not confuse the name of the resource being processed with the name of the resource group from which it was selected.
20	4	**Resource name class address:** address of an 8-byte class name for the currently selected resource. The exit should not change this value.
24	4	**Resource group class address:** address of an 8-byte class name for the resource group from which the current resource was selected. This address is 0 if the resource profile was not built because it is a member of a resource group. The exit should not change this value. Do not confuse the class of the resource being processed with the class of the resource group from which the resource was selected.
28	4	**UACC address:** points to a 1-byte field containing the universal access flags from the resource profile. The possible values are: X'80' - ALTER X'40' - CONTROL X'20' - UPDATE X'10' - READ X'01' - NONE
32	4	**Audit flag address:** points to a 1-byte field containing the audit indicators and qualifiers from the resource profile.
36	4	**Global audit flags address:** points to a 1-byte field containing the global audit indicators and qualifiers from the resource profile.
40	4	**Resource level number address:** points to a 1-byte field containing the LEVEL value from the resource profile.
44	4	**DATA address:** points to a 1-byte length field followed by the installation data that was specified on the RDEFINE or RALTER command for the resource. The length field is zero if no data is present.
48	4	**Access list address:** points to the access list retrieved from the source profile. The first two bytes of the list contain the number of entries in the list. Each entry is nine bytes long: an 8-character userid or group name followed by a 1-byte access authority. The possible values for access authority are: X'80' - ALTER X'40' - CONTROL X'20' - UPDATE X'10' - READ X'01' - NONE

52	4	**Profile anchor address:** points to a working copy of the profile. On the first encounter with a resource, the profile is filled-in with the data taken from the external profile, which is also passed in the preceding five parameters. On subsequent encounters with the resource, the profile is not updated to reflect the data taken from the external profile. It is the responsibility of the exit to modify, if desired, the UACC, audit, global audit, resource level, installation, and application data fields. These modifications have the effect of propagating the first value encountered. On return from this exit, RACF merges access lists according to the value of the rule flags for access list entries.
56	4	**Owner name address:** points to an 8-byte field containing the owner value from the resource profile. If OWNER=YES was specified on the RACLIST request, the owner field has been added to the access list (pointed to by offset 48) with ALTER authority. Once the owner is added to the access list, this information is treated as if it were originally part of the list. This parameter allows the exit to selectively override the effect of the OWNER parameter.
60	4	**APPLDATA address:** points to a 1-byte length field followed by the application data that was specified on the RDEFINE or RALTER command for the resource. The length field is zero if no data is present. RACLIST does not use this field in its processing. This field is intended for use by installation-written routines managing installation-defined resource classes and resources, to provide additional information to the exit routine.
64	4	**NOTIFY address:** points to an 8-byte area containing the userid of the user to be notified when RACF detects an unauthorized attempt to access a resource protected by this profile

Return Codes - 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 not defined; 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 (hex).

FRACHECK Preprocessing Exit (ICHRFX01)

It is extremely important that the writer of the FRACHECK exit routine be aware of the environment in which the routine will be executing. This routine is ***not*** invoked using standard linkage conventions:

- No save area is provided.
- The execution key is unpredictable.
- The exit might receive control in either supervisor or problem program state.
- The exit might or might not be given control APF-authorized.
- The exit should not issue any SVCs.
- Locks might or might not be held on entry.
- The exit routine must be reentrant.
- The exit routine might be given control in either 24 or 31 bit mode.
- If the FRACHECK routine (ICHRFC00) is placed in the FLPA, the exit must also be in the FLPA.

The exit routine ICHRFX01 receives control before FRACHECK performs any processing. On entry to the exit, the following environment exists:

- Register 15 contains the exit's entry point address.
- Register 14 contains the return address.
- Registers 0, 2, 3, and 4 can be used by the exit as work registers.
- Registers 5 through 13 must not be changed by the exit routine.
- Register 1 points to the parameter list as built by the FRACHECK macro and passed to FRACHECK. No validity checking is performed on the contents of the parameter list. Because the parameter list is not copied by FRACHECK, the exit should not modify its contents or information pointed to by its contents. However, the exit can use the first 15 words of the 16-word work area pointed to by the parameter list. The exit must *not* use the 16th word.

The parameter list passed to the FRACHECK exit is as follows:

Offset	Length	Description
0	1	**Authority flags:** contains the requested access authority. Access authority requested is: X‘02’ - for READ X‘04’ - for UPDATE X‘08’ - for CONTROL X‘80’ - for ALTER *Note:* These bit mappings are identical to the bit mappings for the RACHECK ATTR parameter pointed to from offset 8 in the RACHECK preprocessing and and postprocessing exit parameter list.
1	3	Reserved.
4	4	**ENTITY name address:** points to a field as large as the maximum length name of the given class, as determined by the class descriptor table. Names in the field are left-justified and padded with blanks if necessary.
8	4	**CLASS name address:** points to an 8-byte field containing the class name; the name is left-justified and padded with blanks if necessary.
12	4	**ACEE address:** points to the ACEE that RACF uses for authorization checking. If this address is 0, RACF will use the ACEE pointed to by TCBSENV in the current task control block (TCB) or ASXBSENV in the address space extension block (ASXB).
16	4	**APPL name address:** points to an 8-byte field containing the name of the application requesting authorization checking. This name is left-justified and padded with blanks if necessary. If no application name is specified, the address is 0.
20	4	**WKAREA address:** points to 16 fullwords of storage; the exit can use the first 15 fullwords. Because the FRACHECK preprocessing exit cannot issue SVCs, this area is intended to be used by the exit as a work area or register save area.
24	4	**INSTLN address:** points to the value specified on the INSTLN parameter on the FRACHECK macro instruction. If the INSTLN parameter was not specified, the address is 0. No system modules specify the INSTLN parameter; it is intended for use by installation-written routines that invoke FRACHECK to communicate with the FRACHECK preprocessing exit routine.

Return Codes - 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 (hex).

FRACHECK Postprocessing Exit (ICHRFX02)

The FRACHECK postprocessing exit routine receives control after ICHRFC00 completes processing and before it returns control to the FRACHECK issuer. However, if there is an FRACHECK preprocessing exit (ICHRFX01) and it set a nonzero return code, ICHRFX02 is not called.

This exit has the same set of parameters and the same restrictions on the calling environment as the FRACHECK preprocessing routine. In addition, the 16-word work area pointed to by the parameter list will contain the following information:

- 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.

Return Codes - FRACHECK Postprocessing Exit

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 16.

When the exit returns to ICHRFC00 with a zero return code, the contents of work area word 13 are passed back to the FRACHECK issuer as a return code in register 15; the contents of work area word 14 as a profile address in register 1. ICHRFC00 does not validate the contents of these work area fields before it returns.

Password Encryption Exit (ICHDEX01)

The password encryption exit, ICHDEX01, is called by the RACF manager whenever it is necessary to store or compare encrypted password or OIDCARD data in a user profile. RACF will have enqueued on the RACF data set 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.

This exit is also called by ICHRXT00 when the routine is called via RACXTRT TYPE=ENCRYPT,ENCRYPT=(...,INST).

On entry to ICHDEX01, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Length address:** points to a fullword containing the number of fullwords in this parameter list.
4	4	**Function code address:** points to an area containing the function code. The function 0 means encrypt the data; 4 means compare the data.
8	4	**Data length address:** points to a fullword containing the length of the clear text data and encrypted data fields.
12	4	**Clear text address:** points to an area containing the clear text data (the parameter at offset 8 gives the data length).
16	4	**Encrypted data address:** for the compare function, points to an area containing the encrypted version that is to be compared against the clear text. For the encrypt function, the exit returns the encrypted data to the area pointed to by this address.
20	4	**Template code address:** points to a 1-byte area containing the code describing the template type containing the field being worked on. The possible values are: 1 - Group 2 - User 3 - Connect 4 - Data set 5 - General
24	4	**Template name address:** points to an 8-byte area containing the template name of the field being worked on.
28	4	**Profile name address:** points to an 8-byte area containing (the first part of) the profile name.

Return Codes - Encryption Exit

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 result 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 data encryption standard (DES) algorithm.

Note: If register 15 contains any other value, RACF treats it as a return code of 4.

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 Data Encryption Standard algorithm.
C (12)	The clear text data and the encrypted data should be considered not equal.

Note: If register 15 contains any other value, RACF treats it as a return code of 4.

New Password Exit (ICHPWX01)

The RACINIT SVC and the ALTUSER and PASSWORD TSO commands invoke the ICHPWX01 routine. 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 gains control when the following conditions are true:

- The new password is different from the current password.
- The new password is different from the previous passwords if the password history option is active.
- The new password obeys all of the installation′s syntax rules.

On entry to the ICHPWX01 exit routine, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Length address:** points to a fullword containing the number of fullwords in this parameter list.
4	4	**Caller address:** points to a 1-byte field containing the calling function identity: X‘01’ - RACINIT X‘02’ - PASSWORD Command X‘03’ - ALTUSER Command *Note:* If the caller is RACINIT, the ACEE control block might not be present.
8	4	**CPPL address:** points to the TSO command processor parameter list. This applies only to the PASSWORD and ALTUSER commands. If the TSO command processor parameter list is absent, the address is zero.
12	4	**NEWPASS address:** points to an area of the following format: Offset 0, length 1: Length of new password Offset 1, length 8: New password If ENCRYPT = NO was specified, the password is treated as if it is already encrypted. If a new password is not specified, the address is zero.
16	4	**INTERVAL address:** points to a 4-byte field containing the desired password interval from the PASSWORD command. If this interval is absent, the address is zero.
20	4	**Userid address:** points to an area of the following format: Offset 0, length 1: Length of userid Offset 1, length 8: Userid
24	4	**Exit work area address:** points to a fullword whose contents are either: — Zero, for ALTUSER and PASSWORD commands — The contents of the user work address that RACINIT processing passes to ICHRIX01 and ICHRIX02.
28	4	**Current password address** points to an area of the following format: Offset 0, length 1: Length of current password Offset 1, length 8: Current password If ENCRYPT = NO was specified, the password is treated as if it is already encrypted.
32	4	**Password Last Change Date Address:** points to a 3-byte area that contains the date of the last password change. The format of this area is: yyddds where: ‘yy’ is the year ‘ddd’ is the day ‘s’ is the packed decimal sign.
36	4	**ACEE address:** points to the ACEE used. This address may not be available if the caller is RACINIT.
40	4	**Group name address:** points to a 9-byte structure containing a 1-byte length field, followed by an 8-byte field containing the connect group name.

44	4	**Installation data address:** points to an area containing the installation parameters. This address is only available when the caller is RACINIT and the INSTLN parameter was specified.
48	4	**Password history address:** points to an area containing the user's password history. The passwords are in masked or encrypted format, with the oldest password first in the list. The format of the area is: a 2-byte count of the entries in the list, and for each entry a 1-byte reserved field followed by an 8-byte field containing the encrypted password. The SETROPTS PASSWORD(HISTORY(n)) option controls the number of past keywords that are kept.
52	4	**Flag byte address:** points to a 1-byte field containing the form of the current and new passwords: X'00' — Clear text form X'01' — Encrypted form (If ENCRYPT=NO is specified on RACINIT, the password is treated as if it is already encrypted.)

This parameter is available only if the caller is RACINIT.

In all cases, if a parameter is not present, its address is zero.

The following table shows which fields are available to the exit when called from the different RACF components.

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	Userid	X	X	X
24	Work area	X	X	X
28	Current password	X[1]	-	O[2]
32	Password last change date	X	-	O[2]
36	ACEE	X[3]	X	X
40	Group name	O	-	-
44	Installation data	O	-	-
48	Password history	X	-	O[2]
52	Flag byte	X	-	-

X means always available
O means may be available
— means never available

Notes:

1. *Not available if PASSCHK=NO was specified.*
2. *Available only if NEWPASS is available.*
3. *Although available, the ACEE might not be fully initialized.*

Return Codes - 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 interval value change request 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: If register 15 contains any other values, processing terminates with an ABEND.

Command Preprocessing Exit (ICHCNX00)

The ICHCNX00 exit, invoked by RACF commands and the ICHUT100 utility, 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 pre/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.

On entry to the ICHCNX00 preprocessing exit routine (called from the ADDSD, ALTDSD, DELDSD, PERMIT, LISTDSD, and SEARCH commands, and from the ICHUT100 utility), register 1 contains the address of the following area.

Notes:

1. *Because the parameter list passed to the exit is also passed to the RACDEF preprocessing exit and the RACHECK pre/postprocessing exit, the address of the following area is also contained in:*

   - *The fullword at offset 40 of the parameter list pointed to by register 1 on entry to the RACDEF preprocessing and postprocessing exits*

   - *The fullword at offset 60 of the parameter list pointed to by register 1 on entry to the RACHECK preprocessing and postprocessing exits*

2. *For a rename request, the ICHCNX00 parameter list contains both the old and new names, while RACF processes the naming convention table twice (as a rename of the old name and a define of the new name).*

Offset	**Length**	**Description**
0	4	**Length address:** points to a fullword containing the number of fullwords in this parameter list.
4	4	**Caller address:** points to a 2-byte field containing a function code and subfunction code identifying the caller: X'0100' - RACHECK X'0201' - RACDEF DEFINE X'0202' - RACDEF RENAME X'0203' - RACDEF ADDVOL X'0204' - RACDEF DELETE X'0205' - RACDEF CHGVOL X'0301' - ADDSD SET X'0302' - ADDSD NOSET X'0303' - ADDSD MODEL X'0401' - ALTDSD SET X'0402' - ALTDSD NOSET X'0501' - DELDSD SET X'0502' - DELDSD NOSET X'0601' - LISTDSD prelocate call X'0602' - LISTDSD DATASET X'0603' - LISTDSD ID or PREFI X

X‘0701’ - PERMIT TO-resource
X‘0702’ - PERMIT FROM-resource
X‘0801’ - SEARCH prelocate
X‘0802’ - SEARCH postlocate
X‘0900’ - ICHUT100

8 4 **Authority flag address:** points to a 1-byte field containing the user's authorization to the requested function:

X‘08’ - READ
X‘80’ - ALTER or CREATE

In order to issue the SEARCH command for a data set a user requires at least READ authority. In order to issue LISTDSD for a data set specifying the AUTHUSER or ALL operands, the user must have ALTER authority or the equivalent.

12 4 **Resource name address:** points to a 1-byte field containing the resource name length followed by a 44-byte area containing the resource name. The name is left-justified.

16 4 **Old name address:** points to a 1-byte field containing the length of the name followed by a 44-byte area containing the name of the data set that was renamed. The name is left-justified.

20 4 **Volume serial address:** points to an area containing a 1-byte count field followed by a variable number of 6-byte fields containing volume serial identifiers, each left-justified and padded on the right with blanks.

24 4 **Old volume serial address:** points to a 6-byte area containing the volume serial identifier, left-justified and padded on the right with blanks.

28 4 **Resource class name address:** points to an 8-byte field containing the resource class name (DATASET). See the notes for the LISTDSD and SEARCH commands in Figure 7-2.

32 4 **Qualifier address:** points to an 8-byte field containing the data set qualifier. The qualifier is left-justified and padded on the right with blanks. This value is initialized to the high-level qualifier of the data set with the exceptions noted in Figure 7-2. If the exit changes the value, processing proceeds with the changed value. For ADDSD, RACDEF DEFINE, and RACDEF RENAME, RACF determines if the value is a userid or a group defined to RACF. For the other commands and ICHUT100, RACF determines if the value is a userid.

36 4 **Data set type address:** points to a 1-byte flag field indicating the type of data set:

X‘01’ - unknown
X‘40’ - group data set
X‘80’ - user data set

The use of this field is explained in more detail in the following topic, “Return Codes - Command Preprocessing Exit ICHCNX00.”

40 4 **Authority address:** points to a 1-byte flag field containing the authority granted by the exit:

X‘01’ - None
X‘80’ - ALTER

As noted in Figure 7-2, this field is used only for the LISTDSD command. It is intended for those cases when the exit gives the user the authority to list the data set description, which requires READ authority, but not list the access list, which requires ALTER authority.

44 4 **CPPL address:** points to the command processor parameter list (mapped by the IKJCPPL macro instruction). The CPPL can be used to prompt or send messages to a TSO user. As noted in Figure 7-2, the address is zero in non-TSO cases.

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 Figure 7-2 for a summary of these parameters.

		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	0	0	P
	NOSET	P	P	P	C	0	P[3]	0	P	C	0	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
ICHUT100		P	P	P	C	0	0	0	P	C	0	0	0

P = the field is passed to the exit routine, and should not be changed by the exit routine.
C = the field is passed to the exit routine, and may be changed by the exit routine.
0 = the field is not passed to the exit routine, and is indicated as zero.

Notes:

[1] The field is set to the value specified (or defaulted to) on the DATASET, ID, or PREFIX parameter.

[2] The field is set to the value specified on the MASK parameter, or to zero length if the NOMASK parameter was specified.

[3] The field is nonzero only when the VOLUME parameter was specified.

[4] The field is nonzero only when the FVOLUME parameter was specified.

Figure 7-2. ICHCNX00 Exit Parameter Processing

Return Codes - Command Preprocessing Exit ICHCNX00

When the ICHCNX00 preprocessing exit routine returns control, register 15 should contain one of the following return codes:

Hex (Decimal)	Meaning
0 (0)	Normal processing is to continue.
4 (4)	The request is not accepted, and is to be failed. The failure is to be logged (if logging is in effect), and a message is to be issued.
8 (8)	The request is not accepted, and is to be failed. The failure is to be logged (if logging is in effect), but no message is to be issued. Note, however, that messages 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 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 (for example, logging) is to continue.

If register 15 contains any other value, processing proceeds as if the return code was 0.

Note: The data set type address, located at offset 36 in the parameter list, is zero except for 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.

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 matching the qualifier field (offset 32).
- If the data set type is set to X‘40’, a group profile must exist matching the qualifier field (offset 32).
- If the data set type is set to X‘01’ or to any other value, either a user or 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 group profile must exist, but the command issuer need not have any other authority.

Command Preprocessing Exit (ICHCCX00)

The ICHCCX00 exit, invoked by the RACF commands DELGROUP, DELUSER, and REMOVE, allows an installation to perform additional security checks and to further enhance or restrict the RACF limitations on the passed commands.

On entry to the ICHCCX00 preprocessing exit routine, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Caller address:** points to a 1-byte field identifying the calling command: X'0A' - DELGROUP X'0B' - DELUSER X'0C' - REMOVE
4	4	**Entity address:** points to an 8-byte field containing the entity name; the name is left-justified and padded with blanks. For DELUSER and REMOVE, the field is initialized to the userid; for DELGROUP, the field is initialized to the group name.
8	4	**Search argument address:** points to an area containing a 1-byte length field followed by a 44-byte field containing the search argument for the data set search. For REMOVE and DELGROUP, the value is set to the group name followed by a period; for DELUSER, the value is set to the userid of the user being deleted, followed by a period. Although the exit can change the value, the length should remain within the range of 1 through 44.
12	4	**CPPL address:** points to the command processor parameter list.

Return Codes - 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 was 0.

RACF Report Writer Exit (ICHRSMFE)

On entry to ICHRSMFE, register 1 contains the address of the following area:

Offset	Length	Description
0	4	Reason code address: points to a fullword containing the reason for this call: X'00' - Prerecord selection X'04' - Postrecord selection X'08' - End-of-file
4	4	Data string address: points to an area having the following format: 0 4 Address of DATA string 4 2 Length of DATA string 6 1 1-byte flag field. If bit 0 is on, DATA was specified on the RACFRW command.
8	4	Switch address: points to a 1-byte field containing switches indicating whether the RACF report writer will select or reject the record, based on the existing selection/rejection criteria. The format of these switches is: **Bit** **Description** 0 If 0, the record is selected. For RACF records with reason code = 0, this bit is 0 because no selection criteria has yet been applied. If 1, the record is rejected. For non-RACF records, this bit is 1 because non-RACF records are passed to this exit routine for inspection only. 1 If 0, the record is a reformatted RACF SMF record. For more information about these records, see "Reformatted RACF SMF Records" in Chapter 11. If 1, the record is a non-RACF SMF record. 2-7 Reserved.
12	4	Record address: points to the non-RACF SMF record or the reformatted RACF SMF record under inspection. For reason code X'08', this address is set to zero.
16	4	SYSPRINT DCB address: points to an area containing the SYSPRINT DCB that has been opened. The SYSPRINT DCB parameters are: DSORG=PS, RECFM=FA, MACRF=PM, and LRECL=133.
20	4	Communication area address: points to a fullword communications area that can be used by the exit routine. Initially, this field is set to zero.

ICHRSMFE Return Codes

When the ICHRSMFE exit routine returns control to the RACF report writer, register 15 contains 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 was 0.

MVS Router

The system authorization facility (SAF) provides an installation with centralized control over system security processing by using a system service (it is not a part of the RACF program product) called the MVS router. The MVS router provides a focal point and a common system interface for all products providing resource control. The resource managing components and subsystems call the MVS router as part of certain decision-making functions in their processing, such as access control checking and authorization-related checking. These functions are called "control points." This single SAF interface encourages the use of common control functions shared across products and across systems.

The MVS router is always present, whether RACF is present or not. If RACF is available in the system, the MVS router passes control to the RACF routine (ICHRFR00) that invokes the appropriate RACF function based on the parameter information and the RACF router table (ICHRFR01), which associates router invocations with RACF functions. (For more information on the RACF router table, see "RACF Router Table" in Chapter 5.) Before the RACF routine is called, the MVS router calls an optional, installation-supplied security processing exit if one has been installed.

Control points that issue the RACROUTE macro instruction enter the MVS router in the same key and state as the RACROUTE issuer. Control points that continue to issue the RACF macro instructions (such as RACHECK and RACDEF) go directly to RACF, bypassing the router.

MVS Router Exit

The MVS router provides an optional installation exit that is invoked whether or not RACF is installed and active on the system. If RACF is not available, the router exit may act as an installation-written security processing (or routing) routine. If RACF is available, the exit acts as a RACF preprocessing exit.

The only way to invoke the MVS router exit routine is by issuing the RACROUTE macro instruction. The exit is entered by a branch and link instruction and therefore executes in the same key and state as the issuer of the RACROUTE macro. The exit must be named ICHRTX00 and must be located in the link pack area (LPA). The router passes the parameter list to the exit routine. In addition, the exit receives the address of a 150-byte work area.

Control points that continue to use the RACF macro instructions do not invoke the MVS router exit routine.

On entry to the MVS router exit routine, register 1 contains the address of the following area:

Offset	Length	Description
0	4	**Parameter list address:** points to the MVS router parameter list.
4	4	**Work area address:** points to a 150-byte work area that the exit can use.

Return Codes - MVS Router Exit

The exit routine returns one of the following return codes in register 15:

Hex (Decimal)	Meaning
0 (0)	The exit has completed successfully. Control proceeds to the RACF front end routine for further security processing.
C8 (200)	The exit has completed successfully. The MVS router translates this return code to a return code of 0 and returns control to the issuer of the RACROUTE macro, bypassing RACF processing. (See the note below.)
CC (204)	The exit has completed successfully. The MVS router translates this return code to a return code of 4 and returns control to the issuer of the RACROUTE macro, bypassing RACF processing. (See the note below.)
D0 (208)	The exit has completed processing. The MVS router translates this return code to a return code of 8 and returns control to the issuer of the RACROUTE macro, bypassing RACF processing. (See the note below.)
Other	If the exit routine sets any return code other than those described above, the MVS router returns control directly to the issuer of the RACROUTE macro passing the translated code. Further RACF processing is bypassed. You must ensure that the return codes the exit routine uses are defined by the particular RACF function being invoked.

Notes:

1. *In the case of return codes 200, 204, and 208 (decimal), the exit routine should provide return and reason code information in registers 15 and 0 respectively. The MVS router places the return and reason code information in the first two words of the input parameter list. The return and reason codes should have the same definitions as the return and reason codes of the appropriate RACF macro. (See SPL: Supervisor.)*

2. *In the case of return codes other than return codes 200, 204, or 208 (decimal), the exit should place the return and reason code information in the parameter list.*

JES Early Userid Verification Using the MVS Router Exit

The SETROPTS JES(EARLYVERIFY/NOEARLYVERIFY) command determines whether the MVS router exit gets control. The EARLYVERIFY specification turns on an indicator in the RACF CVT that JES can test. If the indicator is on and JES is reading a batch job that does not qualify for user identification propagation, JES makes a call to the system authorization facility (SAF) using the RACROUTE macro.

The RACROUTE macro is issued with the REQUEST=VERIFY and ENVIR=VERIFY parameters. This invocation of SAF does not result in a call to RACF. It is intended to provide, through the SAF MVS router exit, access to optional userid verification logic that is developed and installed by the installation.

Although many possibilities exist, the exit could validate userids by testing for certain combinations of userid, jobname, and accounting information and issuing the appropriate router exit return code.

JES Handling of the Return Codes from SAF

The default return code of X'04' from SAF, which is always returned when no MVS router exit is installed (or by issuing a return code of X'CC' from the router exit), causes the job to be queued for execution in the normal manner. That is, JES does userid and password validation at initiator time.

When the installation has installed an MVS router exit (and within it logically selected the REQUEST=VERIFY, ENVIR=VERIFY call), a successful return code of X'00' can be returned to JES (by issuing a return code of X'C8' from the router exit). In this case, the job is queued for execution with the bypass password validation indicator set, and password validation is subsequently bypassed during RACF initialization processing.

The MVS router exit can also cause a failure return code of X'08' to be returned to JES (by issuing a return code of X'D0' from the router exit), in which case the job is failed immediately.

Note that if RACF is invoked with the RACINIT macro from the MVS router exit, a RACF ABEND could cause a JES ABEND if there is insufficient storage available for JES subtask recovery termination processing.

MVS Router Parameter List

The MVS router parameter list (mapped by macro ICHSAFP) is generated when the RACROUTE macro is issued and describes the security processing request by providing the request type. If the router exit routine exists, the router passes the parameter list to this exit. If RACF is active, the router uses the request type information to invoke the appropriate RACF function.

Field Name	Offset	Length	Description
SAFPRRET	0	4	**Return code** - Defines the RACF or exit routine return code.
SAFPRREA	4	4	**Reason code** - Defines the RACF or exit routine reason code.
SAFPPLN	8	2	**Length** - Defines the length of the MVS router parameter list.
	10	2	**Reserved.**
SAFPREQT	12	2	**Request type** - A binary halfword corresponding to the REQUEST parameter on the RACROUTE macro. The codes for the REQUEST parameters and the associated request types are: AUTH (RACHECK) - 1 (01) FASTAUTH (FRACHECK) - 2 (02) LIST (RACLIST) - 3 (03) DEFINE (RACDEF) - 4 (04) VERIFY (RACINIT) - 5 (05)
	14	2	**Reserved.**
SAFPREQR	16	4	**Request name address** - Points to an 8-byte character field containing the control point name.
SAFPSUBS	20	4	**Subsystem name address** - Points to an 8-byte character field containing the calling subsystem's name, version, and release level.
SAFPWA	24	4	**SAF work area address** - Points to a 512-byte work area for use by the MVS router and the RACF front end routine.
	28	4	**Reserved.**
	32	4	**Offset** - Contains the (signed) offset from the start of the MVS router parameter list to the RACF parameter list.

Chapter 8. The RACF Data Set

This chapter describes the format of the RACF data set. The RACF data set holds all RACF access control information; thus RACF processing uses the data set extensively. RACF uses the RACF data set each time a RACF-defined user enters a system and each time a user wants to define or access a RACF-protected resource.

Multiple RACF data set support provides an alternative to maintaining all of your RACF profiles on one data set. You can have up to 255 RACF data sets.

Each RACF data set is a non-VSAM single extent data set, made up of 1K blocks, and it must be cataloged. The RACF manager addresses these blocks by relative byte addresses (RBAs). These 6-byte RBAs are converted to TTRs before I/O processing begins. The RACF manager uses EXCPVR to read or write to a data set.

A RACF data set consists of several types of records:

- **Header block (index control block, ICB):** the first block in a RACF data set; provides a general description of the data set.
- **BAM (block availability mask) blocks:** used to determine allocation of space within the RACF data set.
- **Index blocks:** used to locate entity records (profiles) in the RACF data set.
- **Templates:** contain mappings of the entity records for the RACF data set.
- **Profiles (entity records):** descriptions of the attributes and authorities for every entity defined to RACF. These entities are:
  - User profiles
  - Group profiles
  - Connect profiles
  - DASD data set profiles
  - Profiles for resources defined by entries in the class descriptor table

Header Block (ICB)

The ICB has a relative byte address of 0. Basically, RACF uses the ICB to locate the other blocks in a RACF data set. Each RACF data set has an ICB, but RACF only uses the ICB for the primary data set when determining the setting of options. The ICB contains the following information:

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	structure	1024	INVICB	Index control block
0	(0)	character	372	ICBHDR	
0	(0)	A-address	4	ICBCHAIN	Reserved
4	(4)	signed	4	ICBBAMNO	Number of BAM blocks in data set
8	(8)	bitstring	6	ICCIBRBA	RBA of highest level index block

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
14	(E)	bitstring	6	ICISSRBA	RBA of first block of index sequence set (See “Index Blocks.”)
20	(14)	bitstring	6	ICBAMRBA	RBA of the first BAM block
26	(1A)	bitstring	1	ICBFLAGS	Status
26	(1A)	1...		ICBEXTND	RACF data set has been extended. RACF will no longer process updates to this version except for statistics updating.
26	(1A)	.1..		ICBUPDAT	VSAMDSET group must be updated
26	(1A)	..11 1111			Reserved
27	(1B)	A-address	1	ICTMPCNT	Number of templates
28	(1C)	bitstring	6	ICBAMHWM	BAM high water mark (See “BAM blocks.”)
34	(22)	character	160	ICBTEMP	Space for 10 template definitions, 16 bytes each in the following format:
34	(22)	signed.	2	ICTMPL	Template length
36	(24)	A-address	1	ICTMPN	Template number
37	(25)	bitstring	1	ICTMRSV1	Reserved
38	(26)	bitstring	6	ICTMPRBA	RBA of template
44	(2C)	character	6	ICTMRSV2	Reserved
194	(C2)	character	1	ICBSTAT	Status
194	(C2)	1...			Reserved
194	(C2)	.1..		ICBNLS	Bypass RACINIT statistics
194	(C2)	..1.		ICBNDSS	Bypass data set statistics
194	(C2)	...1		ICBNTVS	Bypass tape volume statistics
194	(C2)	.... 1...		ICBNDVS	Bypass direct access volume statistics
194	(C2)	.... .1..		ICBNTMS	Bypass terminal statistics
194	(C2)	.... ..1.		ICBNADS	Bypass ADSP protection
194	(C2)	1			Reserved
195	(C3)	bitstring	1	ICBEXTA	Reserved
196	(C4)	character	1	ICBSTAT1	Tape and DASD volume protection options
196	(C4)	1...		ICBTAPE	Tape volume protection is in effect

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
196	(C4)	.1..		ICBDASD	DASD volume protection is in effect
196	(C4)	..1.		ICBDGEN	Generic profile checking in effect for DATASET class
196	(C4)	...1		ICBDGCM	Generic command processing in effect for DATASET class
196	(C4)	.... 1...		ICBRDSN	Input data set name used for logging and messages
196	(C4)	.... .1..		ICBJXAL	JES-XBMALLRACF is in effect
196	(C4)	.... ..1.		ICBJCHK	JES-EARLYVERIFY is in effect
196	(C4)	1		ICBJALL	JES-BATCHALLRACF is in effect
197	(C5)	character	1	ICBAUOP	RACF audit options
197	(C5)	1...			Reserved
197	(C5)	.1..		ICBAGRO	Log GROUP class
197	(C5)	..1.		ICBAUSE	Log USER class
197	(C5)	...1		ICBADAT	Log DATASET class
197	(C5)	.... 1...		ICBADAS	Log DASDVOL class
197	(C5)	.... .1..		ICBATAP	Log TAPEVOL class
197	(C5)	.... ..1.		ICBATER	Log TERMINAL class
198	(C6)	bitstring	1	ICBAXTA	Reserved
199	(C7)	character	1	ICBEROP	Miscellaneous options
199	(C7)	1...		ICBTERP	Perform terminal authorization checking
199	(C7)	.1..		ICBTUAC	Default UACC for terminals not defined to RACF. If on, UACC=NONE. If off, UACC=READ.
199	(C7)	..1.		ICBAVIO	Do not log RACF command violations
199	(C7)	...1		ICBSAUD	Do not log special user activity
199	(C7)	.... 1...		ICBMGDG	Model (GDG) in effect if on
199	(C7)	.... .1..		ICBMUSR	Model (USER) in effect if on
199	(C7)	.... ..1.		ICBMGRP	Model (GROUP) in effect if on
199	(C7)	1		ICBLGRP	List-of-groups access checking effective if on

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
200	(C8)	signed	40	ICBCHCT	Change array for all levels of index blocks
240	(F0)	A-address	1	ICBPINV	Maximum password interval value (1-254)
241	(F1)	bitstring	4	ICBCSTA	Class statistics mask. Bits, when on, indicate that the class should have statistics recorded. The class corresponding to each bit position is defined by the class descriptor entries (POSIT=).
245	(F5)	bitstring	4	ICBCAUD	Class audit mask. Bits, when on, indicate that the class should be logged. The class corresponding to each bit position is defined by the class descriptor entries (POSIT=).
249	(F9)	bitstring	4	ICBCPRO	Class protection mask. Bits, when on, indicate that protection is in effect for the class. The class corresponding to each bit position is defined by the class descriptor entries (POSIT=).
253	(FD)	A-address	1	ICBPHIST	Password history generation value (1-31). Zero if not active.
254	(FE)	A-address	1	ICBPRVOK	Password verification attempt revoke value (1-31). Zero if not active.
255	(FF)	A-address	1	ICBWARN	Password expiration warning level value (1-255). Zero if not active.
256	(100)	character	80	ICBPSYN	Password syntax rules (RULE1 through RULE8).
336	(150)	A-address	1	ICBINACT	Userid inactive interval value (1-255). Zero if not active.
337	(151)	bitstring	4	ICBCGEN	Generic profile checking class mask

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
341	(155)	bitstring	4	ICBCGCM	Generic command processing class mask
345	(159)		4	ICBMOPT	Miscellaneous options
345	(159)	bitstring	1		
345	(159)	1...		ICBFPDS	Global access checking in effect for DATASET class
345	(159)	.1..	1	ICBTDSN	Tape data set protection in effect
345	(159)	..1.			Reserved
346	(15A)	1...	1	ICBPRO	Protect-all in effect
346	(15A)	.1...		ICBPROF	Protect-all (warning) in effect
346	(15A)	.0...			Protect-all (failure) in effect. This flag is ignored if ICBPRO=0
346	(15A)	..1.		ICBEOS	Erase-on-scratch in effect
346	(15A)	...1		ICBEOSL	Erase-on-scratch by SECLEVEL in effect. This flag is ignored if ICBEOS=0
346	(15A)	.... 1...		ICBEOSA	Erase-on-scratch in effect for all data sets. This flag is ignored if ICBEOS=0
346	(15A)	.... .111			Reserved
347	(15B)	1...	1	ICBPROG	Program control in effect
347	(15B)	.111 1111			Reserved
348	(15C)	bitstring	1		Reserved
349	(15D)	bitstring	4	ICBFPTH	Global access checking class mask
353	(161)	binary	1	ICBSLVL	Security level for erase-on-scratch
354	(162)	binary	2	ICBRETP	Security retention period for tape data sets
356	(164)	unsigned	1	ICBQLLN	Length of prefix for single-level data set names
357	(165)	character	9	ICBQUAL	Prefix for single-level data set names
366	(16E)	character	2		Reserved
368	(170)	signed	4	ICBCHCTD	Change count for data blocks. Used to maintain current in-storage blocks with a shared RACF data set.
372	(174)	character	8	ICBSWPW	Encrypted password for RVARY SWITCH command
380	(17C)	character	8	ICBINPW	Encrypted password for RVARY ACTIVE/INACTIVE command
388	(184)	character	636	ICBRSVD	Reserved

BAM Blocks

Each 1K BAM block contains header information followed by block masks. The BAM blocks determine the availability of all the blocks in a RACF data set.

Header

The header contains the following information:

- **Bytes 1-6:** The RBA of the previous BAM block (or 0 if this is the first block).
- **Bytes 7-12:** The RBA of the next BAM block (or 0 if this is the last block).
- **Bytes 13-18:** The RBA of the first block whose space this BAM block defines.
- **Bytes 19-20:** The number of blocks whose space this BAM block defines. (A maximum of 2008 blocks can be defined by one BAM block.)

Block Masks

Each byte of the block masks defines the status of two RACF data set blocks. Each bit corresponds to a 256-byte segment within a block. If a bit is set to 1, the corresponding segment is free. In Figure 8-1, only the first two segments are in use.

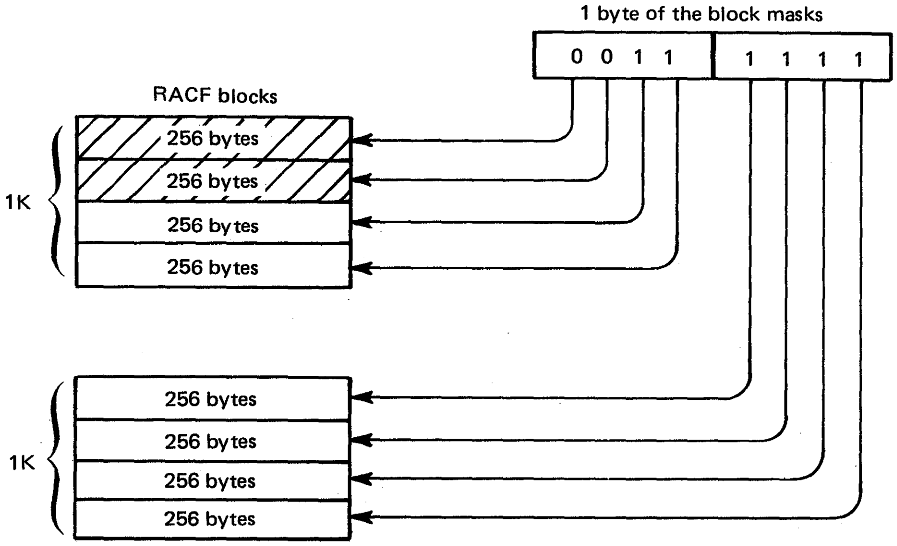


Figure 8-1. Block Masks

The BAM high water mark contained in the ICB is the address of the BAM block from which segments were last allocated or unallocated. This BAM block is used first when starting a search for space for a new allocation.

Index Blocks

RACF uses a multilevel index to locate profiles in a RACF data set. All index searches begin with the highest level index block, whose RBA is contained in the ICB. At every level but the lowest, the first entry in a block that is equal to or alphabetically greater than the requested profile name is used to reach the next level index block. If no entry is greater than or equal to the profile name, the index search continues with the RBA pointed to by the last index entry in the block being searched. This situation can occur because of previous index entry deletions.

The lowest level of index blocks (level 1) is known as the sequence set. Index entries in the sequence set contain the RBAs of the actual profiles. The ICB contains the RBA of the first block in the sequence set. Each block then points to the next block in succession.

Figure 8-2 is an example of part of a RACF index. (The level 1 blocks for the first and third level 2 blocks are not shown in the figure.) The figure illustrates the path RACF uses to find the profile named FAL.N.

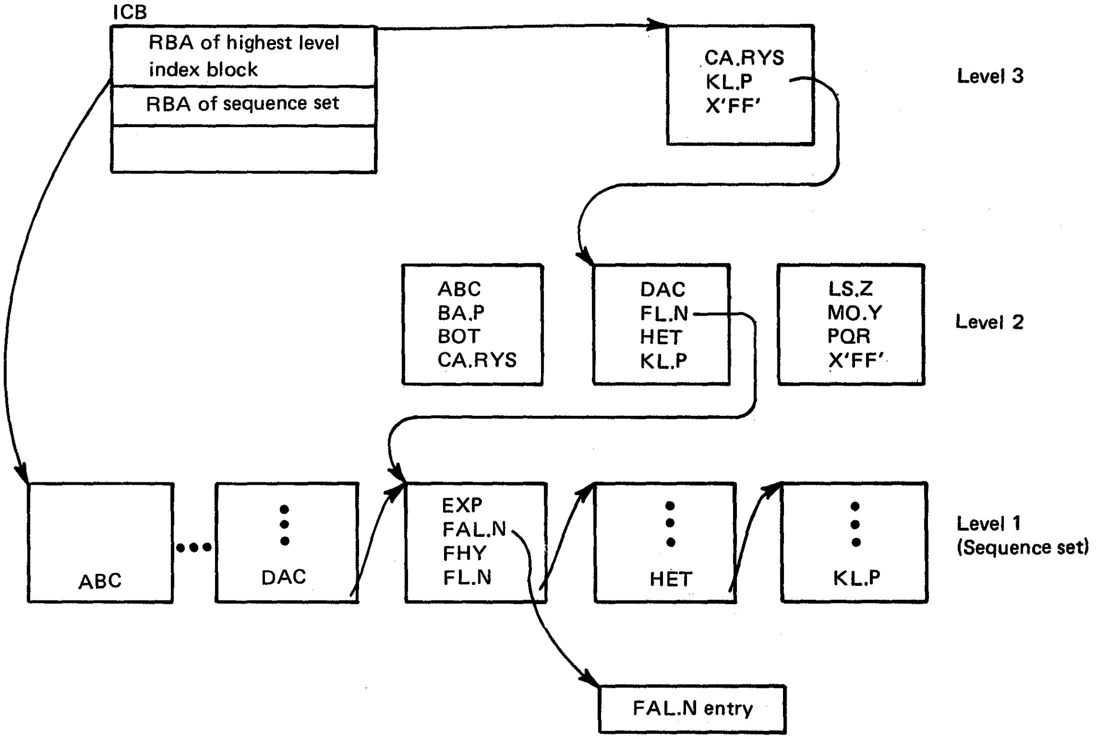


Figure 8-2. Example of a RACF Index

Index Block Format

Each index block contains a 10-byte header followed by the index entries. Level one index blocks also have sequence set chain pointers, which follow the last entry in the block. At the end of each block is a 1-byte end-of-block delimiter, which is set to X'0C'.

Header

The format of the header is:

Byte 1: The index block identifier (X'8A')

Bytes 2-3: The length of the index block (X'0400')

Byte 4: An index block identifier (X'4E')

Byte 5: Format identifier (X'00')

Byte 6: The index level number

Bytes 7-8: The offset to the last entry in the index block

Bytes 9-10: The offset to free space in the index block

Index Entries

The format of each index entry in the block is:

- A 1-byte entry identifier:

  X'21' - for all unique entries, for all index blocks not in level 1, and for the last entry of all index entries in level 1 index blocks that specify the same name

  X'22' - for all index entries in level 1 index blocks that specify the same name except the last entry

- One byte containing the length of the entry name plus 3

- One byte containing the front-end compression count

- The entry name

- A 1-byte entry pointer identifier (X'62' or X'66')

- Six bytes containing the RBA of the next level index block or, for level 1 blocks, the RBA of the profile

Sequence Set Chain Pointers

The format of the sequence set chain pointer in each level 1 index block is:

Byte 1: The chain pointer identifier (X‘20’)

Byte 2: The entry pointer identifier (X‘62’ or X‘66’)

Bytes 3-8: The RBA of the next level 1 block

Index Entry Compression

In order to save space on the RACF data set, index entries are compressed. There are three types of compression:

- Front-end compression
- Last-qualifier encoding
- Upper-level index name compression.

Front-End Compression

Front-end compression leaves the first entry in an index block unchanged. In all other entries of the block, the compression removes the leftmost characters that are identical to the leftmost characters of the first entry. The index entry will contain a count of the number of characters removed from each entry name.

For example, given the following index block:

```
FEG.ABC
FEG.ADE
FEG.F
FEX.P
FOM
GES.B
```

RACF compresses the entries of the block, preceded by the compression counts, as follows:

0	FEG.ABC
5	DE
4	F
2	X.P
1	OM
0	GES.B

If an entry in an index block has exactly the same name as the first entry in the block, the entry may be totally compressed unless the entry is encoded. If the entry is encoded, all but the encode byte will be compressed.

Last-Qualifier Encoding

Last-qualifier encoding is the method of compression used on standard TSO data set names. Last-qualifier compression replaces qualifiers such as .OBJ, .DATA, and .LOAD by hexadecimal 1-byte values that represent the qualifier.

The hexadecimal encode values and the qualifiers that they represent are:

Value	Qualifier
X'01'	.DATA
X'02'	.CNTL
X'03'	.CLIST
X'04'	.LOAD
X'05'	.ASM
X'06'	.FORT
X'07'	.OBJ
X'08'	.PLI
X'09'	.PLIF
X'0A'	.TEXT
X'0B'	.IPLI
X'0C'	.LIST
X'0D'	.STEX
X'0E'	.BASIC
X'0F'	.COBOL
X'10'	.FORTE
X'11'	.FORTG
X'12'	.FORTH
X'13'	.FORTGI
X'14'	.GOFORT
X'15'	.VSBASIC
X'16'	.OUTLIST
X'17'	.LINKLIST
X'18'	.LOADLIST
X'19'	.TESTLIST

Note: The first entry in an index block is not encoded.

Upper-Level Index Name Compression

When a name is added to an upper-level (not level one) index block, RACF compresses the name by eliminating the rightmost characters that are not essential to finding the correct index block at the next lower level.

Templates

Initially, member ICHTEMP0 of the partitioned data set SYS1.MACLIB contains the mappings of the profiles for a RACF data set. When a RACF data set is created, the ICHMIN00 utility program places these mappings, known as templates, in the RACF data set. RACF supplies a template for each type of profile (user, group, data set, general resource, and connect). Each template contains a number that corresponds to the type of profile it is mapping. The template maps the fields that are contained in the profile by describing such items as the field name and field length.

Format of Field Definitions

The templates contain a 14-byte definition for each field in the profile. Each field definition contains information about the field in the following format:

Bytes 1-8: Field name.

Byte 9: Flag field. The bits have the following meanings when they are turned on:

- Bit 0: The field is a member of a repeat group.
- Bit 1: The definition describes a combination field.
- Bit 2: The field is a flag byte.
- Bit 3: The field contains the count of members in the repeat group following this field.
- Bit 4: The definition describes a combination field continued in next entry
- Bit 5: The field is masked.
- Bits 6-7: Reserved.

Bytes 10-12: Reserved.

Bytes 13-14: Field length.

A repeat group consists of one or more sequential fields in a profile that can be repeated. A field that belongs to a repeat group is only defined once in the template, but can be repeated as many times as necessary within the actual profile. A count field precedes the repeat group in the profile indicating how many of these groups follow.

If a field in a profile has a fixed length, a value (less than 255) in the field definition within the template specifies its actual length. If a field in a profile has a variable length, the value in the field definition is 255. The actual field length is contained in the first byte of the data mapped by the field definition.

Templates also contain definitions called combination field definitions. Combination field definitions do not actually describe a field of a profile. They contain the field numbers that reference the field definitions within the template. The fields described by these definitions comprise the combination field; a reference made to the field name of a combination field definition affects all the fields pointed to by the definition.

The format of a combination field definition is different from a non-combination definition. Its format is as follows:

Bytes 1-8: Field name.

Byte 9: Flag field. The flag field always has bit 1 turned on, because the definition describes a combination field.

Bytes 10-14: Reference field numbers of non-combination field definitions that make up combination fields in the template. (The field number references the number of the field in the template and is not a byte offset.)

Group Template

The group template describes the fields of group profiles in the RACF data set.

The contents of the group template are as follows:

Template					Field Being Described
Field Reference Number	**Field Name**	**Flags**	**Rsv'd**	**Field Length**	
1	GROUP	00	00	000	
2	ENTYPE	00	00	001	The number (1) corresponding to group profiles.
3	VERSION	00	00	001	Version number.
4	SUPGROUP	00	00	008	The superior group to this group.
5	AUTHDATE	00	00	003	The date the group was created.
6	AUTHOR	00	00	008	The owner of the group.
7	INITCNT	00	00	002	Reserved.
8	UACC	20	00	001	The universal group authority. (The authority of a user to the group if the user is not connected to the group.) **Bit** **Meaning When Set** 0 JOIN authority 1 CONNECT authority 2 CREATE authority 3 USE authority 4-7 Reserved
9	NOTRMUAC	20	00	001	Identifies whether or not the user must be authorized by the PERMIT command with at least READ authority to access a terminal. (If not, RACF uses the terminal's universal access authority.) If bit 0 is on, the user must be specifically authorized to use the terminal.
10	INSTDATA	00	00	255	Installation data.
11	MODELNAM	00	00	255	Data set model profile name.
12	GRSVFG05	20	00	001	Reserved.
13	GRSVFG06	20	00	001	
14	GRSVFG07	20	00	001	
15	GRSVFG08	20	00	001	
16	GRSVFG09	20	00	001	
17	GRSVFG10	20	00	001	
18	GRSVFG11	20	00	001	
19	GRSFVG12	20	00	001	
20	GRSVFG13	20	00	001	
21	GRSVFG14	20	00	001	
22	FLDCNT	10	00	002	
23	FLDNAME	80	00	008	
24	FLDVALUE	80	00	255	
25	FLDFLAG	A0	00	001	
26	SUBGRPCT	10	00	002	The number of subgroups of the group.

27	SUBGRPNM	80	00	008	A list of the subgroup names.
28	ACLCNT	10	00	002	The number of users connected to the group.
29	USERID	80	00	008	The userid of each user connected to the group.
30	USERACS	A0	00	001	The group authority of each user connected to the group. **Bit Meaning When Set** 0 JOIN authority 1 CONNECT authority 2 CREATE authority 3 USE authority 4-7 Reserved
31	ACSCNT	80	00	002	Reserved.
32	USRCNT	10	00	002	
33	USRNM	80	00	008	Reserved for installation's use.
34	USRDATA	80	00	255	
35	USRFLG	A0	00	001	

Field Name	**Flags**	**Field Number Reference**				
OWNER	40	006	000	000	000	000
DEFDATE	40	005	000	000	000	000
ACL	40	029	030	031	000	000
USERDATA	40	033	034	035	000	000
FIELD	40	023	024	025	000	000

Combination
Fields

Group Profile - Space Requirement

To determine the space required for a group profile in a RACF data set, use the following formula:

```
G = 51 + L + (8 x S) + (11 x U) + I + M
```

where: G = the number of bytes required for a group profile (see note)

L = the number of bytes in the group name

S = the number of subgroups defined to the group

U = the number of users connected to the group

I = the number of bytes in the installation-defined information

M = the number of bytes in the model data set profile name

Note: The RACF data set space required for a group profile is a multiple of the 256-byte segments required to contain the profile. For example, if a group profile contains 352 bytes, the space requirement is 512 bytes.

User Template

The user template describes the fields of the user profiles in a RACF data set.

The contents of the user template are as follows:

Template					Field Being Described
Field Reference Number	**Field Name**	**Flags**	**Rsv'd**	**Field Length**	
1	USER	00	00	000	
2	ENTYPE	00	00	001	The number (2) corresponding to user profiles.
3	VERSION	00	00	001	Version number.
4	AUTHDATE	00	00	003	The date the user was defined to RACF.
5	AUTHOR	00	00	008	The owner of the user profile.
6	FLAG1	20	00	001	Identifies the user as having (bit 0 is on) or not having the ADSP attribute.
7	FLAG2	20	00	001	Identifies the user as having (bit 0 is on) or not having the SPECIAL attribute.
8	FLAG3	20	00	001	Identifies the user as having (bit 0 is on) or not having the OPERATIONS attribute.
9	FLAG4	20	00	001	Identifies the user as having (bit 0 is on) or not having the REVOKE attribute.
10	FLAG5	20	00	001	Identifies the user as having (bit 0 is on) or not having the GRPACC attribute.
11	PASSINT	00	00	001	The interval that the user's password is in effect.
12	PASSWORD	04	00	008	The password associated with the user.
13	PASSDATE	00	00	003	The date the password was last changed.
14	PGMRNAME	00	00	020	The name of the user.
15	DFLTGRP	00	00	008	The default group associated with the user.
16	LJTIME	00	00	004	The time that the user last entered the system by using RACINIT.
17	LJDATE	00	00	003	The date that the user last entered the system by using RACINIT.
18	INSTDATA	00	00	255	Installation data.
19	UAUDIT	20	00	001	Identifies whether or not all RACHECK and RACDEF SVCs issued for the user and all RACF commands (except SEARCH, LISTDSD, LISTGRP, LISTUSER, and RLIST) issued by the user will be logged. If bit 0 is on, they are logged.
20	FLAG6	20	00	001	Identifies the user as having (bit 0 is on) or not having the AUDITOR attribute.
21	FLAG7	20	00	001	Identifies whether or not the user needs to supply a password when entering the system. If bit 0 is on, a password isn't needed.

22	FLAG8	20	00	001	Identifies whether or not the user must supply an operator identification card when logging on to the system. If bit 0 is on, an OIDCARD isn't required.
23	MAGSTRIP	04	00	255	The operator identification associated with the user.
24	PWDGEN	00	00	001	Current password generation number.
25	PWDCNT	10	00	002	Number of old passwords present.
26	OLDPWDNM	80	00	001	Generation number of previous password.
27	OLDPWD	84	00	008	Previous current password.
28	REVOKECT	00	00	001	Count of unsuccessful password attempts.
29	MODELNAM	00	00	255	Data set model profile name.
30	SECLEVEL	00	00	001	User's security level.
31	NUMCTGY	10	00	002	Number of security categories.
32	CATEGORY	80	00	002	List of security categories to which the user has access
33	REVOKEDT	00	00	255	The date the user will be revoked.
34	RESUMEDT	00	00	255	The date the user will be resumed.
35	LOGDAYS	20	00	001	The days of the week the user cannot logon (Bit 0 of this field equals Sunday, bit 1 equals Monday, etc.)
36	LOGTIME	00	00	255	The time of the day the user can logon
37	URSVFG17	20	00	001	
38	FLDCNT	10	00	002	
39	FLDNAME	80	00	008	
40	FLDVALUE	80	00	255	
41	FLDFLAG	A0	00	001	
42	CLCNT	10	00	002	The number of classes in which the user is allowed to define profiles.
43	CLNAME	80	00	008	A list of the classes in which the user is allowed to define profiles. (The user has the CLAUTH attribute.)
44	CONGRPCT	10	00	002	The number of groups that the user is connected to.
45	CONGRPNM	80	00	008	A list of the groups that the user is connected to.
46	USRCNT	10	00	002	Reserved for installation's use.
47	USRNM	80	00	008	
48	USRDATA	80	00	255	
49	USRFLG	A0	00	001	

Field Name	Flags	Field Number Reference				
OWNER	40	005	000	000	000	000
DEFDATE	40	004	000	000	000	000
USERDATA	40	047	048	049	000	000
PASSDATA	40	012	013	000	000	000
NAME	40	014	000	000	000	000
OLDPSWDS	40	026	027	000	000	000
LOGINFO	40	035	036	000	000	000
FIELD	40	039	040	041	000	000

Combination Fields

User Profile - Space Requirement

Use the following formula to determine the space required for a user profile in a RACF data set.

```
U = 98 + L + I + (8 x G) + O + (8 X C) + M + (H x 9)
```

where: U = the number of bytes required for a user profile (see note)

L = the number of bytes in the userid.

I = the number of bytes of installation data (maximum = 255)

G = the number of groups the user is connected to.

O = the number of bytes in the operator ID.

C = the number of classes to which the user has authorization

M = the number of bytes contained in the model profile data set name.

H = the number of generations of old current passwords maintained in the user entry. The maximum value is RCVTHIST from the SETROPTS HISTORY operand.

Note: The RACF data set space required for a user profile is a multiple of 256-byte segments required to contain the profile. For example, if a user profile contains 188 bytes, the profile requires 256 bytes of space.

Connect Template

The connect template describes the fields of the connect profiles in a RACF data set. RACF establishes a connect profile every time a new user is defined to RACF (and connected to a default group) or an existing user is connected to a group. RACF modifies a connect profile when the characteristics of a user connected to a group are modified. The name of the connect profile consists of the user name and the group name separated by a byte of zeroes.

The contents of the connect template are as follows:

Template					Field Being Described
Field Reference Number	**Field Name**	**Flags**	**Rsv'd**	**Field Length**	
1	CONNECT	00	00	000	
2	ENTYPE	00	00	001	The number (3) corresponding to connect profiles.
3	VERSION	00	00	001	Version number.
4	AUTHDATE	00	00	003	The date the user was connected to the group.
5	AUTHOR	00	00	008	The owner of the connect entry.
6	LJTIME	00	00	004	The time that RACINIT was last issued for this user and group.
7	LJDATE	00	00	003	The date that RACINIT was last issued for this user and group.
8	UACC	20	00	001	The default universal access authority assigned to the user for this group. **Bit** **Meaning When Set** 0 ALTER access 1 CONTROL access 2 UPDATE access 3 READ access 4-6 Reserved 7 NONE access
9	INITCNT	00	00	002	The number of RACINIT macro instructions issued for this user and group.
10	FLAG1	20	00	001	Identifies the user as having (bit 0 is on) or not having the ADSP attribute.
11	FLAG2	20	00	001	Identifies the user as having (bit 0 is on) or not having the group-SPECIAL attribute.
12	FLAG3	20	00	001	Identifies the user as having (bit 0 is on) or not having the group-OPERATIONS attribute.
13	FLAG4	20	00	001	Identifies the user as having (bit 0 is on) or not having the REVOKE attribute.
14	FLAG5	20	00	001	Identifies the user as having (bit 0 is on) or not having the GRPACC attribute.
15	NOTRMUAC	20	00	001	Identifies whether or not the user must be authorized by the PERMIT command with at least READ authority to access a terminal. (If not, RACF uses the terminal's universal access authority.) If bit 0 is on, the user must be specifically authorized to use the terminal.

16	CRSVFG07	20	00	001	Reserved.
17	GRPAUDIT	20	00	001	Identifies the user as having (bit 0 is on) or not having the group-AUDITOR attribute.
18	REVOKEDT	00	00	255	The date the user will be revoked.
19	RESUMEDT	00	00	255	The date the user will be resumed.
20	CRSVFG11	20	00	001	Reserved.
21	CRSVFG12	20	00	001	
22	CRSVFG13	20	00	001	
23	CRSVFG14	20	00	001	
24	CRSVFG15	20	00	001	
25	CRSVFG16	20	00	001	
26	FLDCNT	10	00	002	
27	FLDNAME	80	00	008	
28	FLDVALUE	80	00	255	
29	FLDFLAG	A0	00	001	
30	USRCNT	10	00	002	Reserved for the installation's use.
31	USRNM	80	00	008	
32	USRDATA	80	00	255	
33	USRFLG	A0	00	001	

Field Name	Flags	Field Number Reference					
OWNER	40	005	000	000	000	000	Combination Fields
DEFDATE	40	004	000	000	000	000	
USERDATA	40	031	032	033	000	000	
FIELD	40	027	028	029	000	000	

Connect Profile - Space Requirement

The number of bytes required for a connect profile is 49 plus the length of the connect entry name. The size of a connect profile is always less than 256. Therefore, the RACF data set space required for a connect profile is 256 bytes. (A profile requires a multiple of 256-byte segments.)

Data Set Template

The data set template describes the fields of the data set profiles in a RACF data set.

The contents of the data set template are as follows:

Template					**Field Being Described**
Field Reference Number	**Field Name**	**Flags**	**Rsv'd**	**Field Length**	
1	DATASET	00	00	000	
2	ENTYPE	00	00	001	The number (4) corresponding to data set profiles.
3	VERSION	00	00	001	Version number.
4	CREADATE	00	00	003	The date the data set was initially defined to RACF.
5	AUTHOR	00	00	008	The owner of the data set.
6	LREFDAT	00	00	003	The date the data set was last referenced.
7	LCHGDAT	00	00	003	The date the data set was last updated.
8	ACSALTR	00	00	002	The number of times the data set was accessed with ALTER authority.
9	ACSCNTL	00	00	002	The number of times the data set was accessed with CONTROL authority.
10	ACSUPDT	00	00	002	The number of times the data set was accessed with UPDATE authority.
11	ACSREAD	00	00	002	The number of times the data set was accessed with READ authority.
12	UNIVACS	20	00	001	The universal access authority for the data set. **Bit** **Meaning When Set** 0 ALTER access 1 CONTROL access 2 UPDATE access 3 READ access 4-6 Reserved 7 NONE access
13	FLAG1	20	00	001	Identifies whether or not the data set is a group data set. If bit 0 is on, the data set is a group data set.
14	AUDIT	20	00	001	Audit Flags. **Bit** **Meaning When Set** 0 Audit all accesses 1 Audit successful accesses 2 Audit accesses that fail 3 No auditing 4-7 Reserved
15	GROUPNM	00	00	008	The current connect group of the user who created this data set.

16	DSTYPE	20	00	001	Identifies the data set as a VSAM, non-VSAM, MODEL or TAPE data set. **Bit** **Meaning When Set** 0 VSAM data set (non-VSAM if this bit is set to 0) 1 MODEL profile 2 Type = TAPE when set on. 3-7 Reserved
17	LEVEL	00	00	001	Data set level.
18	DEVTYP	00	00	004	The type of device on which the data set resides.
19	DEVTYPX	00	00	008	The EBCDIC name of the device type on which the data set resides.
20	GAUDIT	20	00	001	Global audit flags. (Audit options specified by a user with the AUDITOR attribute.) **Value** **Meaning** x'00' Audit all accesses x'01' Audit successful accesses x'02' Audit accesses that fail x'03' No auditing
21	INSTDATA	00	00	255	Installation data.
22	AUDITQS	00	00	001	Audit SUCCESS qualifier.
23	AUDITQF	00	00	001	Audit FAILURES qualifier.
24	GAUDITQS	00	00	001	Global audit SUCCESS qualifier.
25	GAUDITQF	00	00	001	Global audit FAILURES qualifier. The AUDITQS, AUDITQF, GAUDITQS, and GAUDITQF fields have the following format: **Bit** **Meaning When Set** 0 Log access at READ authority 1 Log access at UPDATE authority 2 Log access at CONTROL authority 3 Log access at ALTER authority 4-7 Reserved
26	WARNING	20	00	001	Identifies the data set as having (bit 7 is on) or not having the WARNING attribute.
27	SECLEVEL	00	00	001	Data set security level.
28	NUMCTGY	10	00	002	The number of categories.
29	CATEGORY	80	00	002	A list of categories to which this data set belongs.
30	NOTIFY	00	00	255	User to be notified when access violations occur against a data set protected by this profile.
31	RETPD	00	00	255	The number of days protection will be provided for the data set. If used the field will be a two-byte binary number.
32	ACL2CNT	10	00	002	The number of program/user combinations currently authorized to access the data set.
33	PROGRAM	80	00	008	The name of each program currently authorized to access the data set.
34	USER2ACS	80	00	008	Userid or group
35	PROGACS	80	00	008	The access authority of the program/user combination.
36	PACSCNT	80	00	002	Access count
37	ACL2VAR	80	00	255	Reserved
38	FLDCNT	10	00	002	

39	FLDNAME	80	00	008	
40	FLDVALUE	80	00	255	
41	FLDFLAG	A0	00	001	
42	VOLCNT	10	00	002	The number of volumes containing the data set.
43	VOLSER	80	00	006	A list of the serial numbers of the volumes containing the data set.
44	ACLCNT	10	00	002	The number of users and groups currently authorized to access the data set.
45	USERID	80	00	008	The userid or group name of each user or group authorized to access the data set.
46	USERACS	A0	00	001	The access authority that each user or group has for the data set. **Bit** **Meaning When Set** 1 ALTER access 1 CONTROL access 2 UPDATE access 3 READ access 4-6 Reserved 7 NONE access
47	ACSCNT	80	00	002	The number of times the data set was accessed by each user or group.
48	USRCNT	10	00	002	
49	USRNM	80	00	008	Reserved for installation's use.
50	USRDATA	80	00	255	
51	USRFLG	A0	00	001	

Field Name	**Flags**	**Field Number Reference**					
OWNER	40	005	000	000	000	000	Combination
DEFDATE	40	004	000	000	000	000	Fields
UACC	40	012	000	000	000	000	
VOLUME	40	043	000	000	000	000	
ACL	40	045	046	047	000	000	
USERDATA	40	049	050	051	000	000	
ACL1	40	045	046	000	000	000	
FIELD	40	039	040	041	000	000	
ACL2	40	033	034	035	036	037	
ACL2A1	40	033	034	035	000	000	
ACL2A2	40	033	034	035	036	000	

Data Set Profile - Space Requirement

Use the following formula to determine the space required for a data set profile in a RACF data set.

```
D = 80 + L + (6 x V) + (11 x U) + I
```

where: D = the number of bytes required for a data set profile (see note)

L = the number of bytes in the data set name

V = the number of volumes indicated for the data set in the profile

U = the number of users and groups in the access list for the data set

I = the number of bytes in the installation-defined information (maximum = 255)

Note The RACF data set space required for a data set profile is a multiple of 256-byte segments required to contain the profile. For example, if a data set profile contains 332 bytes, the profile requires 512 bytes of space.

General Template

The general template describes the fields of general resource profiles in a RACF data set.

The contents of the general template are as follows:

Template					Field Being Described
Field Reference Number	Field Name	Flags	Rsv'd	Field Length	
1	GENERAL	00	00	000	
2	ENTYPE	00	00	001	The number (5) corresponding to profiles for resources defined in the class descriptor table.
3	VERSION	00	00	001	Version number.
4	CLASTYPE	00	00	001	The class to which the resource belongs (from the ID=class-number operand of the ICHERCDE macro).
5	DEFDATE	00	00	003	The date the resource was defined to RACF.
6	OWNER	00	00	008	The owner of the resource.
7	LREFDAT	00	00	003	The date the resource was last referenced.
8	LCHGDAT	00	00	003	The date the resource was last updated.
9	ACSALTR	00	00	002	The number of times the resource was accessed with ALTER authority.
10	ACSCNTL	00	00	002	The number of times the resource was accessed with CONTROL authority.
11	ACSUPDT	00	00	002	The number of times the resource was accessed with UPDATE authority.
12	ACSREAD	00	00	002	The number of times the resource was accessed with READ authority.
13	UACC	20	00	001	The universal access authority for the resource. **Bit** **Meaning When Set** 0 ALTER access 1 CONTROL access 2 UPDATE access 3 READ access 4-6 Reserved 7 NONE access
14	AUDIT	20	00	001	Audit flags. **Bit** **Meaning When Set** 0 Audit all accesses 1 Audit successful accesses 2 Audit accesses that fail 3 No auditing 4-7 Reserved
15	LEVEL	20	00	001	Resource level.

16	GAUDIT	20	00	001	Global audit flags. **Bit** **Meaning When Set** 0 Audit all accesses 1 Audit successful accesses 2 Audit accesses that fail 3 No auditing 4-7 Reserved
17	INSTDATA	00	00	255	Installation data.
18	AUDITQS	00	00	001	Audit SUCCESS qualifier.
19	AUDITQF	00	00	001	Audit FAILURES qualifier.
20	GAUDITQS	00	00	001	Global audit SUCCESS qualifier.
21	GAUDITQF	00	00	001	Global audit FAILURES qualifier. The AUDITQS, AUDITQF, GAUDITQS, and GAUDITQF fields have the following format: **Value** **Meaning** x'00' Log access at READ authority x'01' Log access at UPDATE authority x'02' Log access at CONTROL authority x'03' Log access at ALTER authority
22	WARNING	20	00	001	Identifies the data set as having (bit 7 is on) or not having the WARNING attribute.
23	RESFLG	20	00	001	Resource profile flags: **Bit** **Meaning when set** 0 TAPEVOL may only contain one data set. 1 TAPEVOL profile is automatic. 2 Maintain TVTOC for TAPEVOL. 3-7 Reserved
24	TVTOCCNT	10	00	002	The number of TVTOC entries.
25	TVTOCSEQ	80	00	002	The file sequence number of tape data set.
26	TVTOCCRD	80	00	002	The date the data set was created.
27	TVTOCIND	A0	00	001	Data set profiles flag: **Bit** **Meaning when set** 1 Discrete data set profile exits 2-7 Reserved
28	TVTOCDSN	80	00	255	The RACF internal name.
29	TVTOCVOL	80	00	255	The volumes on which the tape data set resides.
30	TVTOCRDS	80	00	255	The name used when creating the tape data set.
31	NOTIFY	00	00	255	The user to be notified when access violations occur against resource protected by this profile.
32	LOGDAYS	20	00	001	The days of the week the TERMINAL may not be used.
33	LOGTIME	00	00	255	The time of the day the TERMINAL may be used. (Bit 0 equals Sunday, bit 1 equals Monday, etc).
34	LOGZONE	00	00	255	The time zone in which the terminal is located
35	NUMCTGY	10	00	002	Number of categories
36	CATEGORY	80	00	002	List of categories
37	SECLEVEL	00	00	001	Resource security level
38	GRSVFG16	20	00	001	Reserved
39	GRSVFG17	20	00	001	Reserved
40	GRSVFG18	20	00	001	Reserved
41	FLDCNT	10	00	002	
42	FLDNAME	80	00	008	
43	FLDVALUE	80	00	255	

44	FLDFLAG	A0	00	001	
45	APPLDATA	00	00	255	Application data.
46	MEMCNT	10	00	002	The number of members.
47	MEMLST	80	00	255	The resource group member.
48	VOLCNT	10	00	002	Number of volumes in tape volume set.
49	VOLSER	80	00	006	Volume serials of volumes in tape volume set.
50	ACLCNT	10	00	002	The number of users and groups currently authorized to access the resource.
51	USERID	80	00	008	The userid or group name of each user or group authorized to access the resource.
52	USERACS	A0	00	001	The access authority that each user or group has for the resource. **Bit** **Meaning When Set** 1 ALTER access 1 CONTROL access 2 UPDATE access 3 READ access 4-6 Reserved 7 NONE access
53	ACSCNT	80	00	002	The number of times the resource was accessed by each user or group.
54	USRCNT	10	00	002	
55	USRNM	80	00	008	Reserved for installation's use.
56	USRDATA	80	00	255	
57	USRFLG	A0	00	001	

Field Name	Flags	Field Number Reference					
CREADATE	40	005	000	000	000	000	Combination
AUTHOR	40	006	000	000	000	000	Fields
TVTOC	48	025	026	027	028	029	
	40	030	000	000	000	000	
LOGINFO	40	032	033	034	000	000	
ACL	40	051	052	053	000	000	
USERDATA	40	055	056	057	000	000	
ACL1	40	051	052	000	000	000	
FIELD	40	042	043	044	000	000	

General Profile - Space Requirement

To determine the space required for a general profile in a RACF data set, use the following formula:

```
G = 68 + L + I + A + (6 x V) + (11 x U) + (N x M)
```

where: G = the number of bytes required for a general resource profile (see note)

L = the number of bytes in the resource name

I = the number of bytes of installation data (maximum = 255)

A = the number of bytes of application data (maximum = 255)

V = the number of volumes in the tape volume set (for tape volumes only)

U = the number of users and groups in the access list for the resource

N = the number of members in the resource group (resource groups only)

M = the average length of the member name (resource groups only)

Note: The RACF data set space required for a general entity profile is a multiple of 256-byte segments required to contain the profile. For example, if a data set profile contains 342 bytes, then the profile requires 512 bytes of space.

Reserved Templates

Five unused templates are defined for future use. The installation must leave this space reserved and not use it.

The contents of the reserved templates are:

Template					**Field Being Described**
Field Reference Number	**Field Name**	**Flags**	**Rsv'd**	**Field Length**	
1	RSVTMP0x	00	00	000	x can range from 2 to 6
2	ENTYPE	00	00	001	The number corresponding to the type of profile being described

Profiles

The profiles, or entity records, contain the actual descriptions of the attributes and authorities for every entity (users, groups, DASD data sets, and resource classes defined in the class descriptor table) defined to RACF. The connect profiles contain information for each connection made between a user and a group. The number in the entry type field identifies the type of profile and corresponds to the number of the template that maps this type of profile.

Each profile is preceded by a header containing the following information:

- Byte 1: The profile record identifier (X'82')
- Bytes 2-3: The physical record length
- Bytes 4-5: The logical record length
- Byte 6: The length of the record name
- The record name.

The record data (including the type of profile it is) follows the header. This data consists of the fields that are mapped by a template. See the template description corresponding to each type of profile for the contents of these fields.

Because there can be duplicate entry names in different classes, the RACF manager will add a class identifier to the beginnings of entry names (for example, DASD-, TAPE-, or TERM- for DASD volumes, tape volumes, or terminals, respectively). Generic profile names will have the first period in a DATASET profile replaced by X'01', and the dash in the class identifier for general resource classes replaced by X'02'. Although these expanded names are transparent to the user, they will appear when using the block update utility command.

When a tape volume profile is initially created, RACF places the tape volume serial in the volume list of the profile. RACF creates an index entry and profile name in the standard way. If another tape volume is to be added (thus creating a volume set), RACF adds its volume serial to the volume list in the profile and creates an index entry for the volume that points to the profile. (For example, if there are 6 tape volumes in a tape volume set, there will be 6 index entries pointing to the same profile and 6 volume serials in the profile's volume list.) When a tape volume is deleted, RACF removes the volume serial from the volume list in the profile and deletes the index entry. The profile name does not change, even if the volume after which the profile may have been named is deleted. Therefore, it is possible to have, for example, a profile name of TAPE-TAPE01 without having a corresponding index entry and without having TAPE01 in the volume serial list.

Chapter 9. Storage Estimates

This chapter provides information for estimating the RACF storage requirements for:

- RACF data set
- System libraries
- Interactive System Productivity Facility (ISPF) data sets
- RACF Version 1 Release 7 (virtual storage)

RACF Data Set Storage Requirement

This section describes two methods for estimating the size of a RACF data set. The first method is an approximation, the second is a detailed formula. Note that when a RACF data set becomes full, you can extend it with the split/merge/extend utility program (ICHUT400).

Approximation of the RACF Data Set Size

The direct access space needed for a RACF data set 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 and how efficiently the space within the RACF data set is utilized.

On the average, a RACF data set requires approximately 320,000 bytes for each 1000 entities defined to RACF.

Detailed Formula for the RACF Data Set Size

You can use the formula in Figure 9-1 to estimate the size required for a RACF data set.

$N = 11 + A + B + C$

where:

N = the number of 1K (1024-byte) blocks required.

11 = the number of blocks required for the ICB and the IBM defined templates.

A = the number of blocks required for profiles:

$$A = \frac{J}{4} + \frac{K}{4} + \frac{L}{4} + \frac{M}{4} + \frac{N}{4}$$

where:

J = the number of users defined to RACF
K = the number of groups defined to RACF
L = the number of DASD data sets defined to RACF
M = the number of user-group connections
N = the number of general resources defined to RACF

Note: The divisor 4 is used because, in most cases, approximately 4 profiles fit in one 1024-byte block (that is, a profile fits in one 256-byte segment). However, the profile size is variable depending on installation data, the number of users in a group, the number of entries in a data set access list, etc. The information about templates in "Chapter 8: RACF Data Set" can be used to estimate profile size. You may want to replace the divisor 4, in one or more factors, with 3 or less if the profiles require more than one 256-byte segment on the average.

B = the number of index blocks required:

$$B = \sum_{i=1}^{10} (D/E^{i})$$

where:

$D = J + K + L + M + N$

where J, K, L, M, and N are as described above.

E = the number of names that fit in an index block, which is approximately:

$$\frac{1024 \times .5}{10 + \text{average name length}}$$

Note: The formula for E assumes that the index blocks are half full (.5). If you extend an existing RACF data set with the ICHUT400 utility program, you can determine (with the ICHUT200 utility program) how full the index blocks are on the existing data set and replace .5 with the value you determine. In this case, you can also consider the compressed name length when specifying the average name length.

C = the number of BAM blocks required. One BAM block is required for every 2008 blocks in the RACF data set.

$$C = \frac{13 + A + B}{2008}$$

where A and B are as described above.

Note: Round C up to a whole number.

Figure 9-1. Detailed Formula for the RACF Data Set

System Library Storage Requirements

The approximate space requirements for the system libraries (based on the IBM 3350 Disk Storage and a track size of 19,069 bytes) for RACF Version 1 Release 7 are:

Library **SYS1.**	**Number of** **Tracks**	**Contents**
LINKLIB	100	RACF commands, RACF data set initialization program, RACF report writer, data security monitor, RACF class descriptor table, and RACF utilities.
LPALIB	20-22	RACF manager routines, RACF SVC processing routines, RACF RACF service routines, and installation-written exit routines.
MACLIB	21	RACF template definition member (ICHTEMP0), RACF router table macro (ICHRFRTB), RACF class descriptor macro (ICHERCDE), and RACF naming convention macro (ICHNCONV).
HELP	10	Help messages for RACF command and report writer users.
SAMPLIB	5	Sample jobs that you can modify and use for installing RACF.

Note: Add the required tracks for any installation-written exit routines installed for RACF.

See the RACF program directory for the DLIB system library requirements.

Unless you manually change the job stream created by the RACF installation process, the RACF modules will reside in SYS1.LINKLIB and SYS1.LPALIB. If you want the RACF modules to reside in other libraries, keep in mind the following characteristics when selecting the libraries:

- The RACF commands, RACF data set initialization program, data security monitor, and RACF utilities must reside in an APF-authorized library.
- The RACF manager, RACF SVC processing routines, and exit routines run in key 0 and supervisor state, and must reside in the link pack area (LPA, FLPA, or MLPA).

Load module configurations are described in the *RACF Program Logic Manual.*

Interactive System Productivity Facility (ISPF) Storage Requirements

The approximate space requirements for the ISPF data sets (based on the IBM 3350 Disk Storage and a track size of 19,069 bytes) for RACF Version 1 Release 7 are:

ISPF Ddname	Number of Tracks	Contents
ISPPLIB	85	Panel library
ISPSLIB	17	Skeleton library
ISPMLIB	7	Message library
SYSPROC	2	CLIST library

RACF Virtual Storage Requirements

The approximate virtual storage requirements for RACF Version 1 Release 7 are:

Area	Bytes	Comments
FLPA	0	Minimum number of bytes. See Note 1.
PLPA	255,000	See Note 1.
SQA	1300	
LSQA	4200	Need 5200 bytes if the RACF data set is on a shared device. See Note 2. If RACLIST service is used, add Formula 1 for each class. If generic profiles are used, add Formula 2 for each class or data set high-level qualifier. On MVS/XA systems, storage is above 16 megabytes.
CSA		The IBM-supplied default is 15,400 bytes. See Formula 3.
Private Area	8200	Fetch-protected. See Note 2.

Formula 1 = $X + (N \times 16) + M \times (52 + I + A (U \times 9) + (V \times 2)) + O \times (10 + Q + (Q + 1)/2) + P \times (58 + I + (U \times 9) + (V \times 2))$

Formula 2 = $70 + O \times (7 + R) + L \times (51 + I + (U \times 9) + (V \times 2) + (W \times 17))$

where:

X = 24 if you do not use generic profiles

52 if you use generic profiles

N = the number of resources in a given class

M = the number of resident profiles (Effective grouping of resources can reduce this figure because entries for multiple resources point to the same resident profile.)

I = the average length of installation data for the resources in a given class

A = the average length of application data for the resources in a given class

U = the average number of users and groups in the access list of resources in a given class

V = the average number of categories per profile (probably = 0)

O = the number of generic resources

Q = the maximum length of a resource name (as given in the class descriptor table)

P = The number of resident generic profiles

Effective grouping of resources can make this figure less than O (P < O) because entries for multiple resources point to the same resident profile.

R = the average encoded length of a resource name

L = the number of generic profiles actually used for authorization checking

W = the average number of users in the conditional access lists

Formula 3 = 2750 + (1056 x number of resident index/data blocks) + (2200 x number of primary RACF data sets)

On MVS/XA systems, storage is above 16 megabytes. If you are using the resident data blocks option, the portion allocated to resident blocks are in fetch-protected storage.

If you do not use the resident data block option, the number of resident index/data blocks is the lesser of:

- If a data set name table is used, the sum of the resident index block counts in the table; otherwise, the value you used in the ICHSECOP CSECT
- If the data set is not shared, the total number of index blocks in the RACF data set; if the data set is shared, the total number of non-sequence-set index blocks in the RACF data set

If you use the resident data block option, the number of resident index/data blocks is the sum of the resident index/data block counts in the data set name table. If you do not use a data set name table, the number of resident index/data blocks is the value you specified in the ICHSECOP CSECT

Notes:

1. *If RACF is used for IMS/CICS authorization checking, the FRACHECK service routine (ICHRFC00), the RACF router routine (ICHRFR00), and the SAF router routine (ICHSFR00) must be in fixed LPA. In addition, for IMS authorization checking, IMS/VS-to-RACF routines (ICHRGL00, ICHRGL01, and ICHRGL03) must also be in fixed LPA. If RACF is not used for IMS/CICS authorization checking, you can place these modules in pageable LPA and add the estimates for FLPA to those for PLPA by creating or modifying your IEAFxx list in SYS1.PARMLIB. An example of this list can be found in member RACINSTL in SYS1.SAMPLIB.*

   If an exit routine is installed for the FRACHECK service routine, place it in the same virtual storage area as the FRACHECK service routine. Add the size of this routine to FLPA or PLPA accordingly. Also include in PLPA any exit routines and the started procedures replaceable module that you have included in the system. Add the size of these modules to the PLPA requirement.

2. *These LSQA and private area figures apply to each address space while it is executing a RACF function. However, every address space must have 164 bytes of LSQA at all times.*

Chapter 10. RACF Macros

Some of the macros that RACF uses are part of MVS and not part of RACF; installations receive these macros even if they do not install RACF. These macros are:

- **RACDEF** - used to define, modify, or delete resource profiles for RACF.
- **RACHECK** - used to provide authorization checking when a user requests access to a RACF-protected resource.
- **FRACHECK** - used to provide authorization checking when a user requests access to a RACF-protected resource (similar to RACHECK). However, FRACHECK verifies access to only those resources that have RACF profiles brought into main storage by the RACLIST facility.
- **RACINIT** - used to provide RACF user identification and verification.
- **RACLIST** - used to build in-storage profiles for RACF defined resources.
- **RACROUTE** - used to invoke the system authorization facility (SAF) MVS router.
- **RACSTAT** - used to determine if RACF is active and optionally to determine if RACF protection is in effect for a given resource class. The RACSTAT macro can also be used to determine if a resource class name is defined to RACF.
- **RACXTRT** -used to retrieve specified fields from a resource profile or to encrypt data.

RACDEF, RACHECK, RACINIT, RACLIST, RACROUTE, and RACXTRT are fully documented in *SPL: Supervisor* and *SPL: System Macros and Facilities*. RACHECK and RACROUTE are also documented in *Supervisor Services and Macro Instructions*. FRACHECK and RACSTAT are fully documented only in *Supervisor Services and Macro Instructions.*

The macros that are part of the RACF product are:

- **ICHERCDE** - used to generate entries for the resource class descriptor table
- **ICHRFRTB** - used to generate entries in the RACF router table
- **ICHNCONV** - used to create the installation's naming convention table
- **ICHEINTY** - used to locate and/or update profiles on the RACF data set

- **ICHETEST** - used to test for user-specified conditions on selected fields in a profile on the RACF data set

- **ICHEACTN** - used to retrieve and alter specified fields within a profile on the RACF data set

All of these macros are described in this chapter.

ICHERCDE Macro

The ICHERCDE macro generates entries for the resource class descriptor table. The class descriptor table contains information that directs the processing of general resources. The table consists of an entry for each class except USER, GROUP and DATASET. To generate the table, you must specify the macro once for each class. To identify the end of the class descriptor table, you invoke the macro without specifying any operands.

The class descriptor table has two parts. IBM supplies ICHRRCDX, which must not be modified. The installation optionally supplies ICHRRCDE. Both ICHRRCDX and ICHRRCDE reside in SYS1.LINKLIB and are copied to SQA using RACF initialization. Refer to Chapter 5 for instructions on creating ICHRRCDE or see member RACINSTL in SYS1.SAMPLIB.

Note: Installations should also code an ICHRFRTB macro instruction for each entry added to the class descriptor table to be accessed by the RACROUTE macro instruction. See "ICHRFRTB Macro" later in this chapter.

The ICHERCDE macro produces a CSECT for each invocation. If the CLASS operand is present, the CSECT name is the name of the class being defined; otherwise, the CSECT name is ICHRRCDE.

The ICHERCDE macro definition is as follows:

```
[label]  ICHERCDE [CLASS=class-name]
                  [,ID=class-number]
                  [,GROUP=group-class|MEMBER=member-class]
                  [,MAXLNTH=number]
                  [,FIRST=ALPHA|NUMERIC|ALPHANUM|ANY|
                     NONATABC|NONATNUM[
                  [,OTHER=ALPHA|NUMERIC|ALPHANUM|ANY|
                     NONATABC|NONATNUM]
                  [,POSIT=number]
                  [,OPER=YES|NO]
                  [,DFTUACC=ALTER|CONTROL|UPDATE|READ|NONE]
```

CLASS = class-name

Specifies the name of the resource class. The name must be 4 to 8 alphanumeric or national characters; the first character must be alphabetic or national. Note that the first four characters of a class name defined by this macro must be unique from the first four characters of any other class name. Installations must include a national or numeric character in the first four characters of any class names they define in order to guarantee that installation-defined classes do not conflict with IBM-defined classes. In this way, IBM-defined classes should always have unique class names. If this rule is not followed, RACF issues a severity 4 MNOTE warning.

If you specify any options on the ICHERCDE macro, you must specify the CLASS operand.

ID = class-number

Specifies a number from 1 to 255 that is associated with the class name. RACF stores this number in the general profile to indicate the class type. Numbers 1 through 127 are reserved for use by IBM; numbers 128 through 255 are reserved for use by the installation.

If you specify any options on the ICHERCDE macro, you must specify the ID operand.

GROUP = group-class

Specifies the name of the class that groups the resources within the class specified by the CLASS operand. If you omit this operand, RACF does not allow resource grouping for the resource specified by the CLASS operand.

MEMBER = member-class

Specifies the name of the class grouped by the resources within the class specified by the CLASS operand. The class name must be from 1 to 8 alphanumeric characters. When this operand is specified, the class being defined is a resource group.

MAXLNTH = number

Specifies the maximum length of names of resources defined by the CLASS operand. You must specify a number from 1 to 39. If you do not specify MAXLNTH, the maximum length (the default) is 8.

FIRST =

Specifies a character type restriction for the first character of the resource name.

OTHER =

Specifies a character type restriction for the characters of the resource name other than the first character.

ALPHA

Specifies an alphabetic or national character. ALPHA is the default value for both the FIRST and OTHER operand.

NUMERIC

Specifies a digit (0-9).

ALPHANUM

Specifies an alphabetic, national, or numeric character.

ANY

Specifies any character other than a blank, comma, or semicolon.

NONATABC

Specifies an alphabetic character only. National characters and numerics are excluded.

NONATNUM

Specifies an alphabetic or numeric character. National characters are excluded.

POSIT = number

Specifies which bit position in the bitstrings to use as a flag. This bit is used internally, in different bitstrings, to indicate:

- Whether resource access is allowed or prevented (for example, when RACF is inactive)
- Whether auditing is to take place for resources within the class
- Whether statistics are to be maintained for resources within the class.
- Whether generic profile checking is active for the class.
- Whether generic command processing is active for the class.
- Whether global access checking is active for the class.

Specify a number from 0 to 31 to indicate the bit position. The POSIT value does not have to be unique between different classes. If the value is not unique, then two or more classes would share the same bit. Consequently, changing the audit, statistics or protection attribute of one class changes the attribute for all classes with the same POSIT value (bit).

If you specify any operands, you must specify the POSIT operand.

Note: It is recommended that installation-supplied classes have a POSIT value between 25 and 31 to avoid conflict with IBM-supplied classes. RACF issues a severity 4 MNOTE warning if you do not use a value specified in that range.

OPER = <u>YES</u>|NO

Specifies whether RACF is to take the OPERATIONS attribute in account when it performs authorization checking. If YES is specified, RACF considers the OPERATIONS attribute; if NO is specified, RACF ignores the OPERATIONS attribute. YES is the default.

DFTUACC = ALTER|CONTROL|UPDATE|READ|NONE

Specifies the allowed access when you do not specify any access when defining resources within the class. If you omit DFTUACC, RACF uses the default universal access in the ACEE.

ICHERCDE generates the following entry:

Offset	Length	Format	Description
0	2	binary	Length of the entry (0 when you do not specify any operands)
2	1	binary	Class id number
3	8	EBCDIC	Class name, left-justified and padded with blanks
11	8	EBCDIC	Class name of resource group if entry is not the resource group; class name of member resource if entry is a resource group; blanks if no group exists.
19	1	binary	Maximum length of resource name within class
20	1	binary	Syntax of the first character of the resource name **Bit** **Meaning When Set** 0 Alphabetic allowed 1 National allowed 2 Numeric allowed 3 Special allowed 4-7 Reserved
21	1	binary	Syntax of the characters in the resource name other than the first character **Bit** **Meaning When Set** 0 Alphabetic allowed 1 National allowed 2 Numeric allowed 3 Special allowed 4-7 Reserved
22	1	binary	Universal access for defining resources within this class **Bit** **Meaning When Set** 0 ALTER 1 CONTROL 2 UPDATE 3 READ 4-6 Reserved 7 NONE
23	1	binary	Processing flags **Bit** **Meaning When Set** 0 This class is a resource group 1 Use default UACC from ACEE 2 OPERATIONS attribute is to be taken into account during authority checking 3-7 Reserved
24	4	binary	Position within bit strings for CLAUTH, auditing options, statistics options, and RACF inactive options that correspond to this class (in the form of a bit mask)

The class descriptor table supplied by IBM (ICHRRCDX) is as follows:

```
ICHERCDE      CLASS=DASDVOL,
              ID=5,
              MAXLNTH=6,
              FIRST=ALPHANUM,
              OTHER=ALPHANUM,
              POSIT=0,
              OPER=YES
ICHERCDE      CLASS=TAPEVOL,
              ID=6,
              MAXLNTH=6,
              FIRST=ALPHANUM,
              OTHER=ALPHANUM,
              POSIT=1,
              OPER=YES
ICHERCDE      CLASS=TERMINAL,
              ID=7,
              MAXLNTH=8,
              FIRST=ALPHANUM,
              OTHER=ALPHANUM,
              POSIT=2,
              OPER=NO
ICHERCDE      CLASS=APPL,
              ID=8,
              MAXLNTH=8,
              FIRST=ALPHA,
              OTHER=ALPHANUM,
              POSIT=3,
              OPER=NO,
              DFTUACC=NONE
ICHERCDE      CLASS=TIMS,
              ID=9,
              GROUP=GIMS,
              MAXLNTH=8,
              FIRST=ALPHANUM,
              OTHER=ALPHANUM,
              POSIT=4,
              OPER=NO,
              DFTUACC=NONE
ICHERCDE      CLASS=GIMS,
              ID=10,
              MEMBER=TIMS,
              MAXLNTH=8,
              FIRST=ALPHA,
              OTHER=ALPHANUM,
              POSIT=4,
              OPER=NO,
              DFTUACC=NONE
ICHERCDE      CLASS=AIMS,
              ID=11,
              MAXLNTH=8,
              FIRST=ALPHA,
              OTHER=ALPHANUM,
              POSIT=4,
              OPER=NO,
              DFTUACC=NONE
ICHERCDE      CLASS=TCICSTRN,
              ID=12,
              GROUP=GCICSTRN,
              MAXLNTH=13,
              FIRST=ALPHANUM,
              OTHER=ANY,
              POSIT=5,
              OPER=NO,
              FTUACC=NONE
```

```
ICHERCDE   CLASS=GCICSTRN,
           ID=13,
           MEMBER=TCICSTRN,
           MAXLNTH=13,
           FIRST=ALPHANUM,
           OTHER=ANY,
           POSIT=5,
           OPER=NO,
           DFTUACC=NONE
ICHERCDE   CLASS=PCICSPSB,
           ID=14,
           GROUP=QCICSPSB,
           MAXLNTH=17,
           FIRST=ALPHANUM,
           OTHER=ANY,
           POSIT=5,
           OPER=NO,
           DFTUACC=NONE
ICHERCDE   CLASS=QCICSPSB,
           ID=15,
           MEMBER=PCICSPSB,
           MAXLNTH=17,
           FIRST=ALPHANUM,
           OTHER=ANY,
           POSIT=5,
           OPER=NO,
           DFTUACC=NONE
ICHERCDE   CLASS=GMBR,
           ID=16,
           GROUP=GLOBAL,
           MAXLNTH=39,
           FIRST=ANY,
           OTHER=ANY,
           POSIT=6,
           OPER=NO,
           DFTUACC=NONE
ICHERCDE   CLASS=GLOBAL,
           ID=17,
           MEMBER=GMBR,
           MAXLNTH=8,
           FIRST=ANY,
           OTHER=ANY,
           POSIT=6,
           OPER=NO,
           DFTUACC=NONE
ICHERCDE   CLASS=DSNR,
           ID=18,
           MAXLNTH=39,
           FIRST=ALPHANUM,
           OTHER=ANY,
           POSIT=7,
           OPER=NO
ICHERCDE   CLASS=FACILITY,
           ID=19,
           MAXLNTH=39,
           FIRST=ANY,
           OTHER=ANY,
           POSIT=8,
           OPER=NO
ICHERCDE   CLASS=VMMDISK,
           ID=20,
           MAXLNTH=21,
           FIRST=ANY,
           OTHER=ANY,
           POSIT=18,
           DFTUACC=NONE
```

```
ICHERCDE      CLASS=VMRDR,
              ID=21,
              MAXLNTH=17,
              FIRST=ANY,
              OTHER=ANY,
              POSIT=17,
              DFTUACC=NONE
ICHERCDE      CLASS=VMCMD,
              ID=22,
              MAXLNTH=8,
              FIRST=ANY,
              OTHER=ANY,
              POSIT=14,
              DFTUACC=NONE
ICHERCDE      CLASS=VMNODE,
              ID=23,
              MAXLNTH=8,
              FIRST=ANY,
              OTHER=ANY,
              POSIT=16,
              DFTUACC=NONE
ICHERCDE      CLASS=VMBATCH,
              ID=24,
              MAXLNTH=8,
              FIRST=ANY,
              OTHER=ANY,
              POSIT=15,
              DFTUACC=NONE
ICHERCDE      CLASS=SCDMBR
              ID=25 ,
              GROUP=SECDATA,
              MAXLNTH=39,
              FIRST=ANY,
              OTHER=ANY,
              POSIT=9
              OPER=NO
ICHERCDE      CLASS=SECDATA
              ID=26
              MEMBER=SCDMBR,
              MAXLNTH=8,
              FIRST=ALPHA,
              OTHER=ALPHA,
              POSIT=9,
              OPER=NO,
              DFTUACC=NONE
ICHERCDE      CLASS=FCICSFCT
              GROUP=HCICSFCT,
              ID=27,
              MAXLNTH=17,
              FIRST=ALPHANUM,
              OTHER=ANY,
              POSIT=5,
              OPER=NO,
              DFTUACC=NONE
ICHERCDE      CLASS=HCICSFCT
              MEMBER=FCICSFCT,
              ID=28,
              MAXLNTH=17,
              FIRST=ALPHANUM,
              OTHER=ANY,
              POSIT=5,
              OPER=NO,
              DFTUACC=NONE
```

```
ICHERCDE  CLASS=JCICSJCT
          GROUP=KCICSJCT,
          ID=29
          MAXLNTH=16
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
ICHERCDE  CLASS=KCICSJCT
          MEMBER=JCICSJCT
          ID=30,
          MAXLNTH=16
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
ICHERCDE  CLASS=DCICSDCT
          GROUP=ECICSDCT,
          ID=31,
          MAXLNTH=13,
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
ICHERCDE  CLASS=ECICSDCT
          MEMBER=DCICSDCT
          ID=32,
          MAXLNTH=13,
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
ICHERCDE  CLASS=SCICSTST
          GROUP=UCICSTST,
          ID=33,
          MAXLNTH=17,
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
ICHERCDE  CLASS=UCICSTST
          MEMBER=SCICSTST
          ID=34,
          MAXLNTH=17,
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
ICHERCDE  CLASS=MCICSPPT
          GROUP=NCICSPPT,
          ID=35,
          MAXLNTH=17,
          FIRST=ALPHANUM,
          OTHER=ANY,
          POSIT=5,
          OPER=NO,
          DFTUACC=NONE
```

```
ICHERCDE    CLASS=NCICSPPT
            MEMBER=MCICSPPT
            ID=36,
            MAXLNTH=17,
            FIRST=ALPHANUM,
            OTHER=ANY,
            POSIT=5,
            OPER=NO,
            DFTUACC=NONE
ICHERCDE    CLASS=ACICSPCT
            GROUP=BCICSPCT,
            ID=37,
            MAXLNTH=13,
            FIRST=ALPHANUM,
            OTHER=ANY,
            POSIT=5,
            OPER=NO,
            DFTUACC=NONE
ICHERCDE    CLASS=BCICSPCT
            MEMBER=ACICSPCT
            ID=38,
            MAXLNTH=13,
            FIRST=ALPHANUM,
            OTHER=ANY,
            POSIT=5,
            OPER=NO,
            DFTUACC=NONE
ICHERCDE    CLASS=PMBR
            ID=40,
            GROUP=PROGRAM,
            MAXLNTH=8,
            FIRST=ALPHA,
            OTHER=ALPHANUM,
            POSIT=13,
            OPER=NO,
            DFTUACC=NONE
ICHERCDE    CLASS=PROGRAM
            ID=41,
            MEMBER=PMBR,
            MAXLNTH=8,
            FIRST=ALPHA,
            OTHER=ALPHANUM,
            POSIT=13,
            OPER=NO,
            DFTUACC=NONE
```

ICHRFRTB Macro

The ICHRFRTB macro generates entries in the RACF router table. This table controls the action taken by the RACF router ICHRFR00 when invoked by the RACROUTE macro instruction. The router table has two parts. IBM supplies module ICHRFR0X, which must not be modified. The installation optionally supplies module ICHRFR01 so you can add entries for locally defined resource classes or requestor/subsystem combinations.

The IBM-supplied ICHRFR0X module contains one entry for each entry in the class descriptor table, plus one entry each for DATASET and USER. For all entries, the operands REQSTOR and SUBSYS have the default value (all blanks), and the ACTION operand is set to RACF. In addition, several special entries related to program control and tape data set support are present. These entries also have non-blank REQSTOR and SUBSYS values.

Installations should code an ICHRFRTB macro instruction for each entry added to the class descriptor table that will be accessed by the RACROUTE macro instruction.

ICHRFRTB concatenates the values specified for the REQSTOR, SUBSYS, and CLASS operands to form a 24-character string defining the entry. The macro matches these values against the string formed by the values specified on the RACROUTE macro instruction.

The ICHRFRTB macro definition is as follows:

```
[label] ICHRFRTB  [CLASS=class-name]
                  [,REQSTOR=requestor-name]
                  [,SUBSYS=subsystem-name]
                  [,ACTION=NONE|RACF]
                  [TYPE=END]
```

CLASS = class-name
specifies the name of the resource class. You must use the same name that is specified in the corresponding class descriptor table entry. This operand is required unless TYPE = END is specified.

REQSTOR = requestor-name
Specifies the 8-character name to be used, along with CLASS and SUBSYS, to form the 24-character string defining the entry. Installations should begin requestor names with a national character, because IBM-defined requestor names do not begin with a national character. If you do not specify a requestor name, the default is a string of 8 blanks. If you code REQSTOR, you should also code the CLASS operand.

SUBSYS = subsystem-name
Specifies the 8-character name to be used, along with CLASS and REQSTOR, to form the 24-character string defining the entry. Installations should begin subsystem names with a national character, because IBM-defined subsystem names will not begin with a national character. If no subsystem name is specified, it defaults to a string of 8 blanks. This operand should not be coded unless CLASS is also specified.

ACTION =

Specifies the action to be taken for this entry. This operand is required unless TYPE = END is specified.

NONE - specifies that no action is to be taken for this entry.

RACF - specifies that RACF is to be called for this entry.

TYPE = END

Indicates the end of the ICHRFR01 table. You must code TYPE = END on the last ICHRFRTB macro instruction. If TYPE = END is specified, no other operands may be coded.

The RACF router table supplied by IBM (ICHRFR0X) is as follows:

```
ICHRFRTB     CLASS=DATASET,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=USER,
             ACTION=RACF
ICHRFRTB     CLASS=DASDVOL,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=TAPEVOL,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=TERMINAL,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=APPL,
             ACTION=RACF
ICHRFRTB     CLASS=TIMS,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=GIMS,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=AIMS,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=TCICSTRN,
             ACTION=RACF
ICHRFRTB     CLASS=GCICSTRN,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=PCICSPSB,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=QCICSTRN,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=GMBR,
             ACTION=RACF
ICHRFRTB     CLASS=GLOBAL,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=DSNR,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=FACILITY,
             ACTION=RACF
             TYPE=END
ICHRFRTB     CLASS=SCDMBR,
             ACTION=RACF
```

```
ICHRFRTB    CLASS=SECDATA,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=FCICSFCT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=HCICSFCT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=JCICSJCT,
            ACTION=RACF
ICHRFRTB    CLASS=KCICSJCT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=DCICSDCT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=ECICSDCT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=SCICSTST,
            ACTION=RACF
ICHRFRTB    CLASS=UCICSTST,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=MCICSPPT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=NCICSPPT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=ACICSPCT,
            ACTION=RACF
ICHRFRTB    CLASS=BCICSPCT,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=PMBR,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=PROGRAM,
            REQSTOR=PROGMCHK,
            SUBSYS=CONTENTS,
            ACTION=RACF
ICHRFRTB    CLASS=DATASET,
            REQSTOR=CLOSE,
            SUBSYS=OCEOV,
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=DATASET,
            REQSTOR=TAPEOPEN,
            SUBSYS=OCEOV
            ACTION=RACF
            TYPE=END
ICHRFRTB    CLASS=DATASET,
            REQSTOR=TAPERST,
            SUBSYS=RESTART,
            ACTION=RACF
            TYPE=END
```

ICHNCONV Macro

RACF requires a data set name format where the high-level qualifier of a data set name is a RACF-defined userid 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 SVC routines 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.

RACF compares a data set name against each entry in the table until it finds one that matches the name. If RACF does not find a matching entry, the name remains unchanged.

You create a naming convention table, ICHNCV00, by using the ICHNCONV macro. You must assemble the table and linkedit it into SYS1.LPALIB.

The table can have up to 400 naming convention entries and can handle data set names of different formats. Each table entry consists of:

- One ICHNCONV DEFINE macro - to start the naming convention and assign it a name
- Zero or more ICHNCONV SELECT macros - to specify the conditions when the naming convention processes the data set name
- Zero or more ICHNCONV ACTION macros - to convert the name to the standard RACF format or to change any of the modifiable variables
- One ICHNCONV END macro - to terminate the naming convention

At the end of all the naming conventions, a ICHNCONV FINAL macro terminates the table itself.

ICHNCONV DEFINE

An ICHNCONV DEFINE macro starts a naming convention and assigns it a name.

The format of the ICHNCONV DEFINE macro is:

```
[label] ICHNCONV  DEFINE,NAME=convention name
```

DEFINE
Identifies the start of a naming convention. The ICHNCONV DEFINE must start each naming convention and there must be only one per convention.

NAME = convention name

Specifies a unique name that you can use for the convention.

The convention name must be 1 to 8 characters long and follow the rules for symbols in assembler language.

ICHNCONV SELECT

An ICHNCONV SELECT macro specifies the conditions when the naming convention processes the data set name.

The format of the ICHNCONV SELECT macro is:

```
[label] ICHNCONV  SELECT,COND=(condition|compound condition)
```

SELECT

Identifies that this convention has selection criteria.

If the condition on the COND parameter is true, the actions on the ICHNCONV ACTION macros will be processed, and processing will continue as specified on the ICHNCONV END macro. If the condition on the COND parameter is not true, RACF bypasses the ICHNCONV ACTION macros and continues with the next convention in the table.

If an ICHNCONV SELECT macro is not coded, RACF unconditionally processes the actions specified on the ICHNCONV ACTION macros, and continues as specified on the ICHNCONV END macro.

All ICHNCONV SELECT macros for a naming convention must follow the ICHNCONV DEFINE macro and precede any ICHNCONV ACTION macros.

COND = (condition)

Specifies the conditions that have to exist before the naming convention processes the data set name.

The "condition" may be a simple comparison condition of the form:

COND = (variable,operator,operand)

You can also use a "compound condition" formed by linking two or more ICHNCONV SELECT macros with logical AND and OR operators:

COND = (variable,operator,operand,AND)

or

COND = (variable,operator,operand,OR)

If a naming convention contains more than one ICHNCONV SELECT macro, all of the SELECT macros except the last must contain either AND or OR to link it to the following macro. The last (or only) ICHNCONV SELECT cannot have AND or OR specified. RACF evaluates compound conditions in the order specified.

variable

Specifies the variables that the convention can reference. Valid variables are:

- GQ - input qualifiers
- G - input qualifier array subscript
- UQ - output qualifiers
- U - output qualifier array subscript
- QUAL - character qualifier
- QCT - initial number of qualifiers
- NAMETYPE - type of data set
- EVENT - event code
- VOLUME - volume serial numbers
- V - volume serial number array subscript
- VCT - number of volumes
- OLDVOL - volume serial of old volume
- WKX - temporary work variable
- WKY - temporary work variable
- WKZ - temporary work variable
- WKA - temporary work variable
- WKB - temporary work variable
- WKC - temporary work variable
- RACUID - caller's userid
- RACGPID - caller's current connect group

RACF initializes the variables before the first convention. ICHNCONV passes any changes to a variable to subsequent conventions, but only changes made to the variables UQ, QUAL, and NAMETYPE are passed back to the RACF module that called the naming convention table processing module.

You can reference character and hexadecimal variables by substring; for example, (variable,subscript,substring-start,substring-end). If the variable does not accept subscripts or you omit the subscript, you must code a comma to show that the subscript is omitted. Variables cannot be used to define the extents of substrings. For example, (GQ,2,1,3) refers to the first three characters of the second input qualifier; (EVENT,,2,2) refers to the second byte of the event code.

Example: The definition of the data set BOB.SAMPLE.DATASET on volume 111111, when the naming convention table processing module was called during a TSO session when user RACUSR1 was connected to group RACGRP1, would lead to the following set of initial variables:

```
(GQ,1) = BOB
(GQ,2) = SAMPLE
(GQ,3) = DATASET
(GQ,4) to (GQ,22) = blank
(UQ,0) = blank
(UQ,1) = BOB
(UQ,2) = SAMPLE
(UQ,3) = DATASET
(UQ,4) to (UQ,22) = blank
QCT = 3
QUAL = BOB
NAMETYPE = UNKNOWN
EVENT = X'0201'
(VOLUME,1) = 111111
VCT = 1
G, U, V = -1
WKX, WKY, WKZ = 0
WKA, WKB, WKC = blank
OLDVOL = blank
RACUID = RACUSR1
RACGPID = RACGRP1
```

GQ

input qualifiers of the data set name.

G

input qualifier array subscript.

UQ

output qualifiers of the data set name.

U

output qualifier array subscript.

GQ and UQ are arrays containing the qualifiers of the data set name with the high-level qualifier of the name in (GQ,1) and (UQ,1). G and U are halfword variables used to hold subscripts to the GQ and UQ arrays. G and U are initialized to negative one (-1), which is out of the range of valid subscripts.

Initially the input and output qualifiers are identical; but if the contents of the output qualifiers are changed, the new contents are used as the new data set name. Each qualifier is an eight-byte character field padded on the right with blanks. (The field does not include the periods that separate qualifiers.) Initially, (UQ,0) is blank and is reserved for the convention to set as the new high-level qualifier.

If the name produced by the naming conventions table is longer than 44 characters, it is truncated to 44 characters. Thus, the highest possible number of qualifiers (or the highest possible value for the subscripts G and U) is 22.

If you use GQ or UQ in an ICHNCONV SELECT macro without a subscript, RACF tests the condition for each qualifier in turn until the condition is true. The variables G and U are set to the subscript of the qualifier for which the condition was true, G for the conditions using GQ and U for those involving UQ. If the condition is not true, the subscript variable will be negative one (-1).

For all conditions except NE (not equal), the implied linkage is OR; for the NE condition, the implied linkage is AND. For example,

```
SELECT COND=(GQ,EQ,'ABC')     means

SELECT COND=((GQ,0),EQ,'ABC',OR)
SELECT COND=((GQ,1),EQ,'ABC',OR) ...
```

while

```
SELECT COND=(GQ,NE,'ABC')     means

SELECT COND=((GQ,0),NE,'ABC',AND)
SELECT COND=((GQ,1),NE,'ABC',AND) ...
```

You may use any numeric variable as a subscript for GQ or UQ. If RACF encounters an out-of-range subscript (for example, -1 or 23), RACF uses blanks for the comparison.

If GQ or UQ is in an ICHNCONV ACTION macro without a subscript, RACF uses the current value of G or U respectively as the subscript.

QUAL

An eight-byte character qualifier that RACF uses in authority checking to determine if the data set is the user's data set or a group data set.

QUAL is initially the data set high-level qualifier. If the high-level qualifier is not a userid or group name, you should set QUAL to a userid or group name. Setting QUAL, however, is not the same as setting the data set high-level qualifier. QUAL and the high-level qualifier are two separate fields, used for different RACF processing. Therefore, if you change QUAL, you probably want to set (UQ,0) to the same value as QUAL, especially for generic profile names.

QCT

A two-byte binary field containing the initial number of qualifiers in the data set name.

NAMETYPE

Indicates whether the data set is a user or group data set.

NAMETYPE initially has the value UNKNOWN but a convention action may set the value to be USER or GROUP. The three special constant values UNKNOWN, USER, and GROUP may be used to test and set the value of this field. NAMETYPE is available only when the caller is RACDEF

If the convention sets the value to USER or GROUP, RACF ensures that an appropriate user or group exists and fails the RACF or ADDSD if not.

EVENT

A two-byte hexadecimal field containing the event code that is currently passed to the exit routine.

Values that EVENT may have are:

X'0100' - RACHECK
X'0201' - RACDEF DEFINE (RENAME new name)
X'0202' - RACDEF RENAME (OLD name)
X'0203' - RACDEF ADDVOL
X'0204' - RACDEF DELETE
X'0205' - RACDEF CHGVOL
X'0301' - ADDSD SET
X'0302' - ADDSD NOSET
X'0303' - ADDSD MODEL
X'0401' - ALTDSD SET
X'0402' - ALTDSD NOSET
X'0501' - DELDSD SET
X'0502' - DELDSD NOSET
X'0601' - LISTDSD prelocate (see note)
X'0602' - LISTDSD DATASET
X'0603' - LISTDSD ID or PREFIX
X'0701' - PERMIT TO-resource
X'0702' - PERMIT FROM-resource
X'0801' - SEARCH prelocate (see note)
X'0802' - SEARCH postlocate (see note)
X'0900' - ICHUT100

Note: Prelocate means before a profile is located; postlocate means after a profile is located but before it is displayed.

VOLUME

An array of volume serial numbers for volumes containing the data set. Each volume is a six-byte character field.

V

A two-byte variable that contains a subscript to the volume array. V is initialized to -1.

VOLUME is not available for generic data set profiles and is not available from commands if the VOLUME keyword was not specified. An attempt to reference nonexistent volumes (subscript 0 or greater than the number of volumes in the VOLUME array) gives the VOLUME parameter a value of blanks.

If you reference VOLUME in an ICHNCONV SELECT macro without a subscript, RACF tests the condition for each volume in turn until the condition is true. The variable V is set to the subscript of the volume for which the condition was true. If the condition is not true, the subscript variable will be negative one (-1).

For all conditions except NE (not equal), the implied linkage is OR; for the NE condition, the implied linkage is AND. For example,

```
SELECT COND=(VOLUME,EQ,'ABC')     means

SELECT COND=((VOLUME,1),EQ,'ABC',OR)
SELECT COND=((VOLUME,2),EQ,'ABC',OR) ...
```

while

```
SELECT COND=(VOLUME,NE,'ABC')     means

SELECT COND=((VOLUME,1),NE,'ABC',AND)
SELECT COND=((VOLUME,2),NE,'ABC',AND) ...
```

You may use any numeric variable as a subscript for VOLUME. If VOLUME is in an ICHNCONV ACTION macro without a subscript, RACF uses the current value of V as the subscript.

VCT
A two-byte binary field containing the number of volumes in the VOLUME array. If volume information is not available, VCT has a value of zero.

OLDVOL
A six-byte character field containing the volume serial number of the volume that the data set currently resides on. This field is available only during a RACDEF ADDVOL or RACDEF CHGVOL request.

WKX, WKY, WKZ
These are two-byte binary fields that may be used as temporary work variables to save subscripts and other numeric data within and between conventions.

WKA, WKB, WKC
These are eight-byte character fields that may be used as temporary work variables to save qualifiers and other non-numeric data within and between conventions.

RACUID
The caller's userid.

RACGPID
The caller's current connect group.

operator
Specifies the conditional operator: Valid operators are:

EQ - Equal
GT - Greater than
LT - Less than
GE - Greater than or equal
LE - Less than or equal
NE - Not equal

operand

Specifies a variable, a literal, or one of the following special symbols for use with the NAMETYPE variable:

- USER
- GROUP
- UNKNOWN

The operand used should match the length and type of the variable. If the length does not match, RACF performs padding or truncation in the normal manner. If the type does not match, the results are unpredictable.

If operand is specified as a literal, it can be:

- A character string enclosed in quotes
- A decimal number
- A hexadecimal string in the form X'string'

ICHNCONV ACTION

An ICHNCONV ACTION macro changes the value of variables. Use these macros to convert the data set name to the standard RACF format. RACF processes the ACTION macros in sequence.

The format of the ICHNCONV ACTION macro is:

```
[label] ICHNCONV  ACTION,SET=(variable,value)
```

ACTION
Identifies a naming convention action. You can code multiple ICHNCONV ACTION macros.

SET = (variable,value)
Changes the qualifiers of a data set name and other variables.

variable
Specifies the variables that the convention can reference and set. See the preceding description of ICHNCONV SELECT for a description of these variables.

Only the following variable can be set:

- UQ
- QUAL
- G
- U
- V
- NAMETYPE
- WKA, WKB, WKC
- WKX, WKY, WKZ

value

Specifies the value given to the variable. Value can be another variable, a literal, or one of the following special symbols for use with the NAMETYPE variable:

- USER
- GROUP
- UNKNOWN

The value assigned to a variable should match the length and type of the variable. If the length does not match, RACF performs padding or truncation in the normal manner. If the type does not match, the results are unpredictable.

If you specify value as a variable, it can be any of the variables defined in the description of ICHNCONV SELECT.

If value is a literal, it can be:

- A character string enclosed in quotes
- A decimal number
- A hexadecimal string of the form X'string'

ICHNCONV END

An ICHNCONV END macro terminates the naming convention.

The format of the ICHNCONV END macro is:

```
[label] ICHNCONV  END,NEXT=(convention name|'SUCCESS'|'NEXT'|'ERROR'
```

END

Identifies the end of the naming convention. Each convention must have one ICHNCONV END.

NEXT= (convention name|'SUCCESS'|'NEXT'|'ERROR')

Specifies where control goes after this convention executes, if the conditions specified in the ICHNCONV SELECT macros have been met or if there are no ICHNCONV SELECT macros in this convention.

If NEXT=convention name, processing continues with the specified convention and skips intervening conventions in the table. The specified convention must not precede the current convention in the table; otherwise, the RACF request fails.

If NEXT='NEXT', processing continues with the next convention in sequence. If NEXT='NEXT' is coded or defaulted to on the last convention in the table, processing is the same as if NEXT='SUCCESS' was coded.

If NEXT='SUCCESS', then the convention processing routine bypasses further convention processing and returns "a successful name processing" return code to the RACF routine that called it. The RACF routine will continue to process normally using the name returned by the convention processing routine.

If NEXT='ERROR', then the convention processing routine bypasses further processing and returns "an invalid data set name" return code to the RACF routine that called it. The RACF routine will terminate processing and fail the request.

ICHNCONV FINAL

An ICHNCONV FINAL macro terminates the naming convention table. (The naming convention table has only one ICHNCONV FINAL macro.)

The format of the ICHNCONV FINAL macro is:

```
[label] ICHNCONV  FINAL
```

FINAL
Identifies the end of the naming conventions table. There must be only one ICHNCONV FINAL macro in the table and it must be the last entry in the table.

Example of a Naming Convention Table

The following example of a naming convention table illustrates some ways that a table could be coded.

The first convention checks for data sets that are already in the correct RACF format, with a userid or group name in the high-level qualifier or system data sets that start with the characters SYS. This convention bypasses all further checks because no further changes are needed.

```
ICHNCONV DEFINE,NAME=CHECK1
ICHNCONV SELECT,COND=((GQ,1),EQ,RACUID,OR)
ICHNCONV SELECT,COND=((GQ,1),EQ,RACGPID,OR)
ICHNCONV SELECT,COND=((GQ,1,1,3),EQ,'SYS')
ICHNCONV END,NEXT='SUCCESS'
```

This convention checks for data set names that have three or more qualifiers and any qualifier is the user's id. The userid is moved to the start of the name and deleted from its current position. ICHNCONV sets the type indicator and processing continues with convention CHECK4.

```
ICHNCONV DEFINE,NAME=CHECK2
ICHNCONV SELECT,COND=(QCT,GE,3,AND)
ICHNCONV SELECT,COND=(GQ,EQ,RACUID)
ICHNCONV ACTION,SET=(NAMETYPE,USER)
ICHNCONV ACTION,SET=((UQ,0),(GQ,G)
ICHNCONV ACTION,SET=((UQ,G),'  ')
ICHNCONV END,NEXT=CHECK4
```

For all data sets that did not pass the first two conventions the first four characters of the third and fourth qualifiers are concatenated to form a new fifth qualifier. The user's current connect group becomes a high-level qualifier. Processing continues (by default) with the next convention.

```
ICHNCONV DEFINE,NAME=CHECK3
ICHNCONV ACTION,SET=((UQ,0),RACGRIP)
ICHNCONV ACTION,SET=((UQ,5,1,4),(GQ,3,1,4))
ICHNCONV ACTION,SET=((UQ,5,5,8),(GQ,4,1,4))
ICHNCONV ACTION,SET=(NAMETYPE,GROUP)
ICHNCONV END
```

The installation has decided to enforce a standard that all three-qualifier data set names must have a data set type code as the last qualifier. Any qualifiers that are not in the list will cause the name to be rejected.

```
ICHNCONV DEFINE,NAME=CHECK4
ICHNCONV SELECT,COND=(QCT,EQ,3,AND)
ICHNCONV SELECT,COND=((GQ,3),NE,'PLI',AND)
ICHNCONV SELECT,COND=((GQ,3),NE,'DATA',AND)
ICHNCONV SELECT,COND=((GQ,3),NE,'COBOL',AND)
ICHNCONV SELECT,COND=((GQ,3),NE,'ASM')
ICHNCONV END,NEXT='ERROR'
```

The ICHNCONV FINAL macro terminates the table. An assembler END statement is necessary to terminate the assembly.

```
ICHNCONV FINAL
END
```

ICHEINTY, ICHETEST, and ICHEACTN Macros

RACF uses the ICHEINTY, ICHETEST, and ICHEACTN macros to update the various profiles on the RACF data set.

ICHEINTY locates and/or updates the profile.

ICHETEST tests for user-specified conditions on selected fields in the profile.

ICHEACTN retrieves and alters specified fields within the retrieved profile.

Installations planning to use these macros should exercise caution because the ICHEINTY, ICHETEST, and ICHEACTN macros:

- Perform only limited parameter validation. The module issuing these macros must be authorized (supervisor state, system key, or APF-authorized).
- Do not pass control to any exit routines.
- Do not do any logging.

Note: Readers of this section should be familiar with the information contained in Chapter 8, "The RACF Data Set."

ICHEINTY Macro

The ICHEINTY macro provides a direct interface to the RACF data set through the RACF manager. This macro expands to an SVC 132. Its function is to locate and/or update a profile on the RACF data set. You can use the ICHEINTY macro with the ICHETEST and ICHEACTN macros to test and conditionally update fields in RACF profiles.

The format of the ICHEINTY macro definition is:

```
[label] ICHEINTY operation
                 [,TYPE='GRP'|'USR'|'CON'|'DS'|'GEN']
                 [,ENTRY=entry-address]
                 [,CLASS=class-address]
                 [,VOLUME=volume-address]
                 [,ACTIONS=(action-address,.....)]
                 [,TESTS=(test-addr[,AND,test-addr]...)]
                 [,WKAREA=workarea-address]
                 [,NEWNAME=newname-address]
                 [,RBA=rba-address]
                 [,FLDEF=fldef-address]
                 [,OPTIONS=([NOPRO|TESTM|TESTC|ACTION|
                           NOEXEC|FLDEF],...)]
                 [,SMC=YES|NO]
                 [,GENERIC=NO|YES|UNCOND]
                 [,MF=I|L|(E,address)]
```

operation

Specifies the operation that RACF is to perform on the specified profile. The valid operation values are ADD, ALTER, ALTERI, DELETE, DELETEA, FLDEF, LOCATE, NEXT, NEXTC, and RENAME. This operand is positional and is required if you specify MF=I or MF=L.

Some operations are based on assumptions. If a requested operation violates an assumption, the operation fails. The assumptions for each operation are:

ADD

defines entities to RACF by adding entries to the RACF data set. ADD processing:

- Creates a profile for the new entry with fields containing null values. (See "Using ICHEACTN to Alter Data" later in this chapter.)
- Alters field values as specified by associated ICHEACTN macro instructions.
- Provides space for the profile on the RACF data set and writes the profile.
- Creates an index entry for the profile. The index entry points to the written profile.

Some assumptions and considerations regarding ADD are:

- ADD processing assumes that the profile does not already exist.
- ADD processing requires the TYPE and CLASS operands. If the profile already exists (the index contains the profile name), any of the following conditions causes a return code of 8:
  - TYPE is not 'DS'.
  - TYPE is 'DS' and duplicate data set name creation is being inhibited.
  - TYPE is 'DS' and one of the profiles for the entity name contains in its volume list the volume specified by the VOLUME keyword.
- For TYPE='DS', ICHEINTY sets a return code of 60 if VOLUME is not specified and there are multiple profiles for the entity name.
- For TYPE='GEN', you can add the same entity name to any number of different classes. If the class is TAPEVOL, ICHEINTY sets a return code of 60 if a VOLUME is specified but is not an existing TAPEVOL. ICHEINTY creates a profile for a TAPEVOL only when VOLUME is not specified; otherwise, it updates an existing profile. The macro creates an index entry in either case.

ALTER

alters field values in an existing profile on the RACF data set. ALTER processing:

- Locates the profile on the RACF data set.
- Performs the tests that the ICHETEST macros specify, if any macros are present. The TESTS operand on the ICHEINTY macro names the tests to be performed.
- Alters field values as specified by associated ICHEACTN macro instructions.
- Writes the profile back to the primary and backup (if any) RACF data sets.

Some assumptions and considerations regarding ALTER are:

- ALTER processing requires the TYPE and CLASS operands.
- If the profile is too large to be rewritten to the same field in the RACF data set, ICHEINTY allocates new space, writes the profile to the new location, and updates the index entry for the entity to point to the new location.

ALTERI

is similar to the ALTER operation with the following exceptions:

- The updated field must be in place. ICHEINTY sets a return code of 68 if the altered record cannot be rewritten into its original location on the RACF data set.

- The update to the field occurs with only a shared lock on the RACF data set. Therefore, other ALTERI, LOCATE, or NEXT requests can take place simultaneously.

- You can specify the RBA of the level one index block, containing the sequence set pointer to the profile to be altered.

- ALTERI processing writes the profile back to the primary RACF data set only (not to the backup).

DELETE

deletes a profile from the RACF data set. DELETE processing:

- Deletes the index entry for the entity.
- Frees space used for the profile on the RACF data set.

Some assumptions and considerations regarding DELETE are:

- DELETE processing requires the ENTRY, TYPE, and CLASS operands.

- You cannot specify the ACTIONS operand with the DELETE operation.

- For TYPE='DS', ICHEINTY sets a return code of 60 if you specified VOLUME and a profile containing the specified volume in its VOLSER list was not found. ICHEINTY sets a return code of 56 if you do not specify VOLUME and there are multiple profiles for the entity name.

- ICHEINTY deletes the profile for a TAPEVOL only when the volume being deleted is the last in its set. Otherwise, the macro deletes the index entry and removes the volume from the VOLSER list.

DELETEA

Deletes all the members of a TAPEVOL set from the RACF data set. It is similar to the DELETE operation with the following exception:

- If you specify 'TYPE=GEN' and the class is TAPEVOL, ICHEINTY deletes one profile along with the index entry for each volume in the set.

FLDEF

Builds the area that the FLDEF operand uses. The area contains control information and the two lists generated by the TESTS and ACTIONS operands.

Some assumptions and considerations regarding FLDEF are:

- FLDEF creates a separate area for ICHEACTN and ICHETEST pointers, which can be referenced from one or more ICHEINTY macros.

- You can maintain the field definition area with the MF=E form of "ICHEINTY FLDEF."

- When referencing the field definition area from a remote ICHEINTY, specify FLDEF=field-definition-area, and do **not** specify any of the ACTION, FLDEF, TESTC, and TESTM options on the OPTIONS keyword.

LOCATE

Retrieves zero or more fields from an existing RACF profile in the RACF data set. LOCATE processing:

- Locates the profile in the RACF data set.

- Performs the tests that the ICHETEST macros specify, if any macros are present. The TESTS operand on the ICHEINTY macro names the tests to be performed.

- Retrieves field values as specified by associated ICHEACTN macro instructions into the caller-specified work area.

Some assumptions and considerations regarding LOCATE are:

- ICHEINTY sets a return code of 44 if the values being returned are too large for the work area provided.

- For TYPE='DS', ICHEINTY set a return code of 60 if you specify VOLUME and one or more profiles were found for the data set name but none contained the specified volume name in its VOLSER list. ICHEINTY sets a return code of 56 if you do not specify VOLUME and there are multiple profiles for the entity name.

- ICHEINTY sets a return code of 52 if an ICHETEST macro specified by the TESTS operand failed. The LOCATE operation terminates at this point.

NEXT

Retrieves zero or more fields from the profile whose name follows the name specified by the ENTRY operand. The NEXT operation updates the area pointed to by the ENTRY operand with the name of the profile just completed. NEXT processing:

- Locates the profile of the first entity of the specified type that follows the specified entity located in the RACF data set.

- Performs the tests that the ICHETEST macros specify, if any macros are present. The TESTS operand on the ICHEINTY macro names the tests to be performed.

- Retrieves field values as specified by associated ICHEACTN macro instructions into the caller specified work area.

Some assumptions and considerations regarding NEXT are:

- If the entity retrieved has the same name as the entity that follows it in the RACF data set, ICHEINTY sets the duplicate data set name count. The count becomes 2 if it was zero on entry; otherwise, the count increases by one. The count is zero if the entity is not a duplicate of the one that follows it.

- ICHEINTY sets a return code of 44 if the values being returned do not fit into the provided work area.

- For qualified types (data set, general, and connect), the located entity must have the same high-level qualifier as the specified entity. Otherwise, the macro sets a return code of 12.

- For TYPE='DS', if the duplicate data set name count in the work area is not zero, ICHEINTY locates the specified data set name. (That is, if the duplicate data set count equals n, then the macro locates the nth occurrence of the specified name. If there are less than N occurrences of the specified name, the same process occurs as when the duplicate data set count is zero; ICHEINTY locates the profile of the first entity of the specified type that follows the specified entity in the RACF data set.)

- If an ICHETEST macro specified in the TESTS operand failed, ICHEINTY sets a return code of 52 and the NEXT operation terminates.

NEXTC

is similar to the NEXT operation with the following exception:

- For qualified types (data set, general, and connect), ICHEINTY does not make the high-level qualifier check. For unqualified types (group and user), NEXTC processing is identical to NEXT processing.

RENAME

renames a data set entry in the RACF data set.

Some assumptions and considerations regarding RENAME are:

- You must specify the NEWNAME operand.
- RENAME is valid only with TYPE='DS'

TYPE='GRP'|'USR'|'CON'|'DS'|'GEN'

Specifies the type of the entry as GROUP ('GRP'), USER ('USR'), CONNECT ('CON'), DATASET ('DS'), or general resource ('GEN').

The final parameter list the SVC uses as a request to the RACF manager must include a value for TYPE.

ENTRY=entry-address

Specifies the address of a one-byte entry name length field followed by the entry name. The NEXT and NEXTC operations update this field. The area pointed to must allow for 44 bytes of data to be returned.

CLASS=class-address

Specifies the address of an eight-character class name. The class name is valid only with TYPE='GEN' and is ignored for all other types.

VOLUME=volume-address

Specifies the address of a six-character volume identifier. When TYPE='DS', the volume identifier differentiates among data sets with the same name. When the operation is ADD, TYPE='GEN', and the class is TAPEVOL, the volume identifier specifies the name of an existing tape volume set to which the current entry is to be added. In all other cases, ICHEINTY ignores the volume identifier.

ACTIONS=(action-address,.....)

Specifies the address of one or more ICHEACTN macros that determine which profile field(s) the RACF manager is to retrieve or update. See the description of the ICHEACTN macro later in this chapter.

TESTS=(test-addr[,AND,test-addr]...)

Allows some preliminary testing on selected conditions prior to the execution of the operation specified by the ICHEINTY macro. You must specify an odd number of items (including the connector, 'AND'). Each address must be the address of a list built by the ICHETEST macro. See the description of the ICHETEST macro later in this chapter.

WKAREA=wkarea-address

Specifies the address of the area into which the values are to be retrieved. This operand is valid and required only for the LOCATE, NEXT, and NEXTC operations.

NEWNAME=newname-address

Specifies the address of the new name to be assigned to the entity named by the ENTRY operand. The format of the new name is the same as that for the ENTRY operand. This operand is valid only for the RENAME operation.

RBA=RBA-address

Specifies the address of a six-byte relative byte area (RBA) of a level one index that contains the sequence set pointer to the profile to be altered. This keyword is valid only for an ALTERI request.

FLDEF = fldef-address

Specifies a remote list of ACTION/TEST pointers set up by the FLDEF positional parameter.

OPTIONS = ([NOPRO,TESTM,TESTC,ACTION,NOEXEC,FLDEF]...)

Provides more direct control of the code generated by the EXECUTE form of the macro. (This operand is valid only with the EXECUTE form of the macro.) You can specify one or any number of the following subfields:

NOPRO - Does not generate any prologue code; that is, the instructions that set the type of request, such as ADD, by updating the first two bytes of the parameter list, are not generated.

FLDEF - Generates the FLDEF pointer relocation code to point to the list of ACTION and TEST pointers in the ICHEINTY macro expansion.

ACTION - Generates code to set the number of ACTIONs that are to be performed.

TESTC - Generates code to set the number of TESTs that are to be performed.

TESTM - Generates code to set both actual and maximum number of TESTs.

NOEXEC - Does not generate the SVC instruction to invoke the RACF manager. This subfield is useful with the EXECUTE form of the macro to allow partial setup of the parameter list.

SMC = YES|NO

Controls the 'set-must-complete' operation mode of the RACF manager. YES is the default mode of operation.

GENERIC = NO|YES|UNCOND

Informs the RACF manager whether the given entity name is a generic name.

If GENERIC=NO, the RACF manager does not attempt to convert the name specified by the ENTRY operand from external to internal form. GENERIC=NO is the default.

If GENERIC=YES, the RACF manager attempts to convert the name specified by the ENTRY operand from external to internal form. The RACF manager does the conversion only if the entity name contains a generic character (an * or %). If the entity name does not contain a generic character, processing continues without any conversion.

If GENERIC=UNCOND, the RACF manager unconditionally converts the name specified by the ENTRY operand from external to internal form.

For RENAME, the same process applies also to the NEWNAME operand.

MF = I|L|(E,address)

Specifies the form of the macro as either INLINE, LIST, or EXECUTE.

The INLINE form is similar to a STANDARD form, except that it generates code to branch around the parameter list. In the MF=I form, the label names the first location of the parameter list, not the preceding instruction. MF=I is the default.

The LIST form reserves and initializes storage.

The EXECUTE form modifies a list defined elsewhere. If you use the EXECUTE form, you must specify the address of the list to be modified. The address can be an A-type address or register (2 through 12).

Return Codes from the ICHEINTY Macro

The return codes from the ICHEINTY macro, returned in register 15, are:

Hex (Decimal)	Description
0 (0)	The requested operation was successful.
4 (4)	If the reason code=4, a recovery environment could not be established; if the reason code=0, an invalid function code was specified.
8 (8)	An attempt was made to add an entry to the RACF data set but an identical entry already exists.
C (12)	For requests other than NEXT or NEXTC, the specified entry did not exist. For NEXT or NEXTC requests, no subsequent entries satisfied the request.
10 (16)	Reserved.
14 (20)	The RACF data set did not contain enough space to satisfy the request.
18 (24)	An I/O error occurred while accessing the RACF data set.
1C (28)	RACF was not active at the time of the request.
20 (32)	The request type requires a user work area but the area was not provided (the address in the parameter list was 0).
24 (36)	The input parameter list or the associated ACTION and TEST blocks contain an error. When this code is returned, register 0 contains an additional code. The possible codes in register 0 are: 1 - Invalid entry name 2 - Action(s) specified with DELETE or DELETEA 3 - An action specified for an undefined field 4 - Test(s) specified with RENAME 5 - Reserved 6 - Reserved 7 - Incorrect entry type
28 (40)	The maximum profile size (32,600 bytes) has been reached; the profile cannot be expanded.
2C (44)	The user-supplied work area was not large enough to hold all the data returned. The work area is filled with data up to, but not including, the first field that did not fit.
30 (48)	The user-supplied work area was smaller than the minimum amount required (30 bytes).
34 (52)	A test condition specified in the TESTS keyword of the ICHEINTY macro was not met; further processing was suppressed.
38 (56)	You requested an operation on a DATASET type entry that has multiple RACF definitions, but you did not specify a VOLUME to single out a specific entry.

3C (60) For DATASET type entries, you specified a VOLUME that did not exist in the volume list of any entry with the specified name. For TAPEVOL class entries, a request tried to add a new TAPEVOL to a nonexistent tape volume set.

40 (64) You attempted to delete one of the IBM-defined entries (such as SYS1 or IBMUSER) from the RACF data set.

44 (68) An ALTERI request attempted to increase the size of the entry profile being updated.

48 (72) A request to add an entry to the RACF data set would have caused the RACF index to increase to a depth that RACF does not support. The maximum depth is 10 levels.

4C (76) ICHEINTY encountered an invalid index block or read a non-index block when it expected an index block.

50 (80) You made an attempt to update (by a request other than ALTERI) a RACF data set that has been extended (the ‘extended’ bit in the ICB was on).

ICHETEST Macro

The ICHETEST macro tests for user-specified conditions on selected data in a RACF profile. You can use the ICHETEST macro with the ICHEINTY and/or ICHEACTN macros to ensure that a specific requirement is met before processing of the ICHEINTY and/or ICHEACTN macro occurs. Failure to meet the requirements specified on the ICHETEST macro causes further processing of any associated ICHEINTY or ICHEACTN macros to be suppressed.

The format of the ICHETEST macro is:

```
[label] ICHETEST  FIELD=field-name|address
                  ,FLDATA=(length,address)
                  [,COND=EQ|NE|GT|LT|GE|LE|ONES|ZEROS|
                       MIXED]
                  [,ENCRYPT=TEMPLATE|YES|NO]
                  [,MF=L|(E,address)|I]
```

FIELD = field-name|address

Specifies the field-name in the RACF profile whose value is to be tested.

If you use the LIST form of the macro, specify the name of the field. The name must be from 1 to 8 characters long, not enclosed in quotes, and defined in the RACF template. In addition, the field name must not be a combination field name (such as ACL in the group profile).

If you use the EXECUTE or INLINE form of the macro, specify the address of the field to be tested. The address can be an A-type address or register (2 through 12). For EXECUTE and INLINE, you can also specify the field name as a constant (for example, 'OWNER').

FLDATA = (length,address)

Specifies the field containing the value to be tested against.

The length must be greater than zero and less than or equal to the length of field-name in the FIELD operand. For fixed length fields, you can specify a length that is less than the actual length of the field in the profile. For variable length fields, you must specify as the length the actual length of the field in the profile. For flag fields, the length specified is ignored and a one-byte length is assumed.

COND = EQ|NE|GT|LT|GE|LE|ONES|ZEROS|MIXED

Specifies the relationship that must exist between the FIELD and FLDATA values to satisfy the test.

EQ, NE, GT, LT, GE, and LE are valid only for fixed length or variable length fields. They are invalid for flag fields.

ONES, ZEROS, and MIXED are valid only for flag fields.

If you omit this operand, COND = EQ is the default.

ENCRYPT = TEMPLATE|YES|NO

Specifies whether the data specified by FLDATA is to be encrypted or not, before the test is performed. If ENCRYPT = YES, the data is encrypted regardless of whether the template flag associated with the field specifies that it is to be encrypted. If ENCRYPT = NO, RACF does not encrypt the data regardless of the template flag value. If ENCRYPT = TEMPLATE, the template flag determines whether the data is encrypted.

ENCRYPT is ignored if you specify COND as ONES, ZEROS, or MIXED.

MF = L|(E,address)|I

Specifies the form of the macro as either LIST, EXECUTE, or INLINE.

The LIST form reserves and initializes storage. MF = L is the default.

The EXECUTE form modifies a list defined elsewhere. If you use the EXECUTE form, you must specify the address of the list to be modified. The address can be an A-type address or register (2 through 12).

The INLINE form is similar to a STANDARD form, except that it generates code to branch around the parameter list. In the MF = I form, the label names the first location of the parameter list, not the preceding instruction.

Some considerations regarding the ICHETEST macros are:

- You cannot use the ICHETEST macro with an ICHEINTY macro that has the RENAME operation specified.
- A profile can contain repeat groups. A repeat group consists of one or more sequential fields that can be repeated in the profile. By specifying COND = EQ, you can select the occurrence of the repeat group to which the action applies.

  By specifying COND = NE, you can position yourself past the last occurrence of the repeat group. Then you can add a new occurrence to the end of that repeat group with an ICHEACTN macro. (Note that when the ICHEACTN macro refers to a repeat group and more than one ICHETEST macro is specified, the last ICHETEST macro serves to position data retrieval from the profile. Therefore, the last ICHETEST should refer to the same repeat group as the last ICHEACTN; otherwise the retrieved data will be from the last tested field.)
- Tests involving negative numbers cause unpredictable results.
- If a specified address equals zero, ICHETEST makes no test.
- Use COND = EQ or COND = NE to test masked fields. Other comparisons cause unpredictable results.

ICHEACTN Macro

The ICHEACTN macro retrieves or alters data in a specified RACF profile. It builds a parameter list containing the RACF profile field name and, optionally, the address of ICHETEST macros that control the data accessing.

The format of the ICHEACTN macro is:

```
[label] ICHEACTN  FIELD=field-name|address
                  ,FLDATA=(length,address)|'DEL'|'COUNT'
                  [,TESTS=(address[,AND,address]...)]
                  [,RUN=YES|NO]
                  [,GROUP=YES|NO]
                  [,ENCRYPT=TEMPLATE|YES|NO]
                  [,MF=L|(E,address)|I]
```

FIELD = field-name|address

Specifies the field-name in the RACF profile whose value is to be retrieved or updated.

If you use the LIST form of the macro, specify the name of the field. The name must be 1 to 8 characters long, not enclosed in quotes, and defined in the RACF template.

If you use the EXECUTE or INLINE form of the macro, specify the address of the name of the field to be retrieved or updated. The address can be an A-type address or register (2 through 12). For EXECUTE and INLINE, you can also specify the field name as a constant (for example, 'OWNER').

FLDATA = (length,address)|'DEL'|'COUNT'

Updates or deletes data in a specified RACF profile. This operand is valid only when used with the ALTER, ALTERI, ADD and RENAME operations on the ICHEINTY macro.

The address points to a field containing the value that is to replace the value specified by field-name in the FIELD operand. The address can be an A-type address or register (2 through 12).

Length is the length of the replacement field. The length can be an integer constant or register (2 through 12). The length must be equal to the length of the specified data if you code the GROUP = YES operand on this macro or if the specified data field is of variable length; otherwise the length value is ignored. If you omit the length subparameter, the field is deleted.

'DEL' causes field-name in the FIELD operand to be given a null value or causes an occurrence of a repeat group to be deleted, or (if GROUP = YES is coded) deletes all occurrences of a repeat group.

'COUNT' causes field-name in the FIELD operand to be treated as a positive integer and increased by one.

TESTS = (address[,AND,address]...)

Specifies preliminary testing that must occur before any data retrieval or updating takes place. Each address specified must be the address of a list built by an ICHETEST macro. The address can be an A-type address or register (2 through 12). Multiple addresses indicate that all conditions (tests) must be satisfied. If not, RACF suppresses further processing of the macro. If you omit the logical connector 'AND', you must use a comma to indicate its omission.

Note: If GROUP=YES is also coded on the ICHEACTN macro, all tests specified by the TESTS parameter are ignored or bypassed.

RUN=<u>YES</u>|NO

Specifies if a data retrieval or update is to be actually performed. This operand allows you to code an ACTION operand on the ICHEINTY macro without the action being performed for this particular execution. The default is RUN=YES.

GROUP=YES|<u>NO</u>

Specifies whether an update for a repeat group is for a single occurrence of the group or for the entire group, including the repeat count that contains the number of occurrences. If FIELD=field-name contains the name of a repeat group count field and GROUP=YES, ICHEACTN replaces or deletes the entire repeat group, including the count field. So, when coding GROUP=YES in this situation, code the new repeat count in the first two bytes of the FLDATA field, or specify FLDATA='DEL' to delete the entire group. When replacing an entire group, RACF will automatically delete the old group contents if the old repeat group count is nonzero. The default is GROUP=NO.

Note: If GROUP=YES is also coded on the ICHEACTN macro all tests specified by the TESTS parameter are ignored or bypassed.

ENCRYPT=<u>TEMPLATE</u>|YES|NO

Specifies whether the data specified by FLDATA is to be encrypted or not. If ENCRYPT=YES, the data is encrypted regardless of whether the template flag associated with the field specifies that it is to be encrypted. If ENCRYPT=NO, RACF does not encrypt the data regardless of the template flag value. If ENCRYPT=TEMPLATE, the template flag determines whether the data is encrypted.

MF=<u>L</u>|(E,address)|I

Specifies the form of the macro as either LIST, EXECUTE or INLINE.

The LIST form reserves and initializes storage. MF=L is the default.

The EXECUTE form modifies a list defined elsewhere. If you use the EXECUTE form, you must specify the address of the list to be modified. The address can be an A-type address or register (2 through 12).

The INLINE form is similar to a STANDARD form, except that it generates code to branch around the parameter list. In the MF=I form, the label names the first location of the parameter list, not the preceding instruction.

Using ICHEACTN to Retrieve Data

The ICHEACTN macro retrieves data when used with the ICHEINTY macro having a LOCATE, NEXT or NEXTC operand. When using ICHEACTN to retrieve data, you must supply a work area on the ICHEINTY macro into which the retrieved data can be placed. The first fullword of the work area must be the length of the work area (including the first fullword itself). The minimum work area is 30 bytes, even if no data is being retrieved.

The format of the user work area is as follows:

Offset	Length	Description
0	4	Length of entire user area
4	6	RBA return area
A	1	Flags
B	1	Reserved
C	4	Duplicate data set name count
10	8	Reserved
18	4	Length of data returned into work area
1C	variable	Field value return area

Ensure that the storage in the work area from +4 to +1E is initialized to binary zeroes. If the area is not initialized, it can be difficult to determine if the information returned by the RACF manager is present.

If the profile located has a generic name, bit 0 of the flag byte at offset A is on.

An ICHEINTY macro can have several ICHEACTN macros associated with it. For each ICHEACTN macro, the RACF manager returns into the field value return area:

- A two-byte length field. This length field contains the length of the retrieved data for that particular ICHEACTN macro. Note that this two-byte length field does not contain its own length.

- The retrieved data from the RACF profile.

Note that all the fields are byte-aligned.

Some examples of the different field types that the RACF manager can return in the field value return area are:

1. If a condition specified by an ICHETEST macro (that is associated with the ICHEACTN macro) was not satisfied or if the specified field was a repeat field that contained no members, the field value return area would contain all zeroes.

```
| 0000           |
  0              2
```

2. If the field specified is a fixed-length field, the return field contains the length of the field followed by the field value.

data length	field value
0	2

3. If the field specified is a flag field, the return field contains the length of the field (X'0001') followed by a one-byte value.

0001	flag
0	2

4. If the field specified is a variable-length field, the return field contains the length of the field followed by a one-byte length field (that does not include its own length) followed by the field value.

data length	length	field value
0	2	3

5. If the field specified is a combination field, the return area contains the length of all the fields in the combination, followed by a concatenation of values of each of the individual fields in the combination. If a field in the combination is in a repeat group, all the fields in the combination must be in the same repeat group. (Example 6 shows how the RACF manager returns combinations containing fields of a repeat group.)

data length	concatenation of field values
0	2

6. If the field specified is a field in a repeat group or a combination field made up of one or more fields in the same repeat group, the results returned depend upon whether (1) an ICHETEST macro was associated with the ICHEACTN in order to position to a particular occurrence of the repeat group or (2) no ICHETEST macro was associated and all occurrences are implied.

   When an ICHETEST is associated, the format of the result is the same as if the field were not in a repeat group. When no ICHETEST is associated, the result is the two-byte length field followed by the concatenation of the values of every occurrence of the specified field. If the specified field is a combination field, the values of the fields in the combination are first concatenated for each occurrence, then these concatenations are concatenated in the order of their occurrence.

data length	concatenation of occurrences
0	2

Using ICHEACTN to Alter Data

The ICHEACTN macro alters data when used with the ICHEINTY macro having an ADD, ALTER, ALTERI, or RENAME operand. If the conditions specified by the TESTS keyword on the ICHEACTN macro are met, the field specified in the FIELD operand is assigned the value specified in the FLDATA operand. If the specified field in the RACF profile is in a repeat group, then:

- If you specified a test with COND=EQ, the existing occurrence of the repeat group is altered.
- If you specified a test with COND=NE, a new occurrence is added to the end of the repeat group.
- If you did not specify a test, a new occurrence is added to the beginning of the repeat group.

RACF uses the length specified as a subfield of the FLDATA keyword only when you specify GROUP=YES. For fixed length fields, the data length is the field length in the template. For variable length fields, the data length is the first data byte (it does not include its own length). RACF handles combination fields as a succession of fields, either fixed or variable length. If the combination field contains some but not all of the fields in a repeat group, the fields not included are set to null values.

The specification of FLDATA='COUNT' causes the specified fields to be treated as a positive integer and increased by one. If the field specified is variable length or has a fixed length greater than four, RACF ignores the specification and does not modify the field value.

If you specify FLDATA='DEL', the specified field has a null value; that is:

- For a fixed length field that is not in a repeat group, the field is set to binary ones.
- For a flag field that is not in a repeat group, the field is set to binary zeroes.
- For variable length fields that are not in a repeat group, the length of the field is set to zero.
- For fields within a repeat group or a repeat group field name, the entire group is deleted.

If you specify zero as the "address" value, the result is the same as if you had specified FLDATA='DEL', except that for fields in a repeat group, each field in the group is set to a null value (the same as fields not in a repeat group).

Examples of ICHEINTY, ICHETEST, and ICHEACTN Macro Usage

The following examples illustrate some of the functions provided by the ICHEINTY, ICHETEST, and ICHEACTN macros:

1. **Determining if a user is defined to RACF**

```
*         .
*         .
*         .
          LA    15,WEND-W      LENGTH OF WORK AREA.
          ST    15,W           INITIALIZE WORK AREA.
          XC    WR,WR          CLEAR RESERVED AREA.
          ICHEINTY LOCATE,TYPE='USR',ENTRY=USR1,WKAREA=W
          LTR   15,15          R15=0 IF USER DEFINED TO
                               RACF
          BNZ   NOTDEFD
*         .
*         .
*         .
*    DATA AREAS
USR1      DS    AL1            LENGTH OF USERID (1 TO 8)
          DS    CL8            USERID
W         DS    0F
          DS    F              LENGTH OF WORK AREA.
WR        DS    CL24           RESERVED.
          DS    F
WEND      EQU   *              END OF WORK AREA.
```

The ICHEINTY macro identifies the user profile to be located. A return code of 0 in register 15 indicates that the user is defined to RACF. A return code of 12 indicates that the user is not defined. Note that this ICHEINTY macro contains a work area. By also coding an ICHEACTN macro in this example, you can retrieve current field values from this user profile into the work area.

2. **Adding a userid to a data set access list**

```
*        .
*        .
*        .
         ICHEINTY ALTER,TYPE='DS',ENTRY=DSN1,            *
               ACTIONS=AACL
         LTR   15,15             0 RETURNED IF DS IS RACF
                                 DEFINED
         BNZ   DSNOTDEF          DS NOT RACF DEFINED OR
                                 ERROR
         CLI   TUSERID+1,X'00'   WAS USER ALREADY IN LIST
         BNZ   INLIST            YES.  USER WAS IN LIST
                                 ALREADY
*        .
*        .
*        .
*     DATA AREA
AACL     ICHEACTN FIELD=ACL,FLDATA=(11,ACL),             *
               TESTS=TUSERID,MF=L
TUSERID  ICHETEST FIELD=USERID,FLDATA=(8,USER),COND=NE,  *
               MF=L
DSN1     DS    AL1               DATA SET NAME LENGTH
                                 (1 TO 44)
         DS    CL44              DATA SET NAME
ACL      DS    0CL11             ACCESS LIST ENTRY
USER     DS    CL8               USERID TO BE ADDED
USERACS  DS    XL1               ACCESS TO BE GIVEN:
*                                X'80' FOR ALTER
*                                X'40' FOR CONTROL
*                                X'20' FOR UPDATE
*                                X'10' FOR READ
*                                X'01' FOR NONE
ACSCNT   DC    XL2'0000'         ZERO ACCESS COUNT
```

The ICHEINTY macro identifies the data set profile whose access list is to be updated. It also points to an ICHEACTN macro that describes how the profile is to be updated. In this example, RACF adds a userid to the access list.

The ICHEACTN macro, in turn, points to an ICHETEST macro that tests for certain conditions before the profile can be updated. In this example, ICHETEST tests to determine if the specified userid already exists in the access list. (The second byte of the test block at TUSERID is 0 if the userid is not in the access list.) If the userid does not exist, RACF adds the userid (with the specified access authority) to the access list and updates the data set profile. If the userid already exists, no profile update occurs.

3. **Changing the access authority of a user in a data set access list**

```
*        .
*        .
*        .
         ICHEINTY ALTER,TYPE='DS',ENTRY=DSN1,            *
               ACTIONS=AUSRACS
         LTR   15,15             0 RETURNED IF DS IS RACF
                                 DEFINED
         BNZ   DSNOTDEF          DS NOT RACF DEFINED OR
                                 ERROR
         CLI   TUSERID+1,X'00'   WAS USER IN LIST
         BNZ   NOTINLST          NO.  USER WAS NOT IN
                                 LIST
*        .
*        .
*        .
*     DATA AREA
AUSRACS  ICHEACTN FIELD=USERACS,FLDATA=(1,USERACS),     *
               TESTS=TUSERID,MF=L
TUSERID  ICHETEST FIELD=USERID,FLDATA=(8,USER),COND=EQ,  *
               MF=L
DSN1     DS    AL1               DATA SET NAME LENGTH
                                 (1 TO 44)
         DS    CL44              DATA SET NAME
UACC     DS    XL1               ACCESS TO BE GIVEN:
*                                X'80' FOR ALTER
*                                X'40' FOR CONTROL
                                 X'20' FOR UPDATE
                                 X'10' FOR READ
                                 X'01' FOR NONE
```

This example is similar to the previous example. However, if the userid exists in the data set access list, RACF changes that user's access authority to the value specified in USERACS and updates the data set profile. If the userid does not exist, no profile update occurs.

Note that you can use this example to delete a userid from the data set access list by changing the ICHEACTN macro to read:

```
AUSRACS  ICHEACTN FIELD=USERID,FLDATA='DEL',             *
               TEST=TUSERID,MF=L
```

Chapter 11. SMF Records

RACF produces two SMF records:

- Type 80 - produced during RACF processing
- Type 81 - produced at the completion of RACF initialization

The first 18 bytes of each record represent the standard SMF header. See *System Management Facilities (SMF)* for information about how to use SMF.

For sorting purposes, the RACF report writer reformats SMF records (types 20, 30, 80, and 81) and uses these reformatted records as input to the modules that produce the RACF reports. There are two types of reformatted records - reformatted process records and reformatted status records. If you want to use the RACF report writer exit (ICHRSMFE) to produce additional reports or to add additional record selection criteria, you should familiarize yourself with the layouts of these reformatted records.

Record Type 80 - RACF Processing Record

RACF (module ICHRAU00) writes record type 80 for the following detected events:

- **Unauthorized attempts to enter the system** - During RACF processing of the RACINIT macro instruction, RACF found that a RACF-defined user either (1) has supplied an invalid password, OIDCARD, or group name, or (2) is not authorized access to the terminal.

  RACF always writes this violation record when it detects the unauthorized attempt; this violation record supplements the information that RACF sends to the security console in RACF message ICH408I.

- **Authorized accesses or unauthorized attempts to access RACF-protected resources** - During RACF processing of the RACHECK or RACDEF macro instruction, RACF found that one of the following events occurred:

  1. The user was permitted access to a RACF-protected resource and allowed to perform the requested operation.

  2. The user did not have sufficient access or group authority to access a RACF-protected resource, or supplied invalid data while attempting to perform an operation on a RACF-protected resource.

In the first case, RACF writes the violation record if the ALL or SUCCESS logging option is set in the resource profile by the ADDSD, ALTDSD, RALTER, or RDEFINE command and the access type is within the scope of the valid access types. RACF also writes the record if logging has been unconditionally requested by the RACHECK postprocessing exit routine.

In the second case, RACF writes the violation record if the ALL or FAILURES logging option is set in the resource profile by the ADDSD, ALTDSD, RALTER, or RDEFINE command, or if logging is unconditionally requested by the RACHECK postprocessing exit routine. The violation record supplements the information that RACF sends to the security console in RACF message ICH408I.

Note that the FAILURES (READ) option is the default in cases where new resources are RACF-protected.

For the preceding events, a RACHECK exit routine can modify the logging options by changing the LOG parameter on the RACHECK macro instruction from ASIS to NOFAIL, NONE, or NOSTAT, or by unconditionally requesting or suppressing logging with the logging control field. (For information on the LOG parameter of the RACHECK macro instruction, see *SPL: Supervisor* or *SPL: System Macros and Facilities*. For information on the logging options of the ADDSD, ALTDSD, ALTUSER, RALTER, RDEFINE, and SETROPTS commands, see the *Command Language Reference*.

- **Authorized or unauthorized attempts to modify profiles on a RACF data set -** During RACF command processing, RACF found that a user with the AUDITOR attribute specified that the following be logged:

  1. All detected changes to a RACF data set by RACF commands and the RACDEF SVC

  2. All RACF commands (except LISTDSD, LISTGRP, LISTUSER, RLIST, and SEARCH) issued by users with the SPECIAL attribute

  3. All violations detected by RACF commands (except LISTGRP, LISTUSER, and SEARCH)

  4. All RACHECK and RACDEF SVCs issued for the user and all RACF commands (except LISTGRP, LISTUSER, and SEARCH) issued by the user

  In the first three cases, RACF writes records if a user with the AUDITOR attribute specified AUDIT, SAUDIT, and CMDVIOL, respectively, on the SETROPTS command. In the fourth case, RACF writes the records if a user with the AUDITOR attribute specified UAUDIT on the ALTUSER command.

You can use SMF records to:

- Track the total use of a sensitive resource (if the ALL option is set)

- Identify the resources that are repeated targets of detected unauthorized attempts to access them (if the ALL or FAILURES option is set)

- Identify the users who make detected unauthorized requests

- Track SPECIAL user activity

- Track activity of a particular user

RACF writes one record for each event. (RACF can write two records for one operation on a resource - for example, when a RACF-protected DASD data set is deleted with scratch.)

SMF record 80 contains the following information:

- The record type
- Time stamp (time and date)
- Processor identification
- Event code and qualifier (explained in Table 1)
- User identification
- Group name
- A count of the relocate sections
- Authorities used to successfully execute commands or access resources
- Reasons for logging
- Command processing error flag
- Foreground user terminal ID
- Foreground user terminal level number
- Job log number (job name, entry time, and date)

(The data in a relocate section is explained in Tables 2 and 3.)

RACF provides a RACINIT option to log successful signons and signoffs as well as ENVIR=CREATE or ENVIR=DELETE signons and signoffs. For the LOG keyword on the RACROUTE and RACINIT macros, LOG=ALL or LOG=ASIS may be specified to control the generation of log records for RACINIT. The value of the LOG keyword is passed to both the RACINIT preprocessing and postprocessing installation exits. Both exits are invoked prior to the generation of a log record, and the LOG keyword value can be changed for both exits. The log record RACF creates is a standard type 80 SMF record.

The format of record type 80 is:

Offset Dec(Hex)	Field Name	Length	Format	Source	Description
0(0)	SMF80LEN	2	binary	-	Total record length
2(2)		2	binary	-	Reserved
4(4)	SMF80FLG	1	binary	-	System indicator: **Bit Meaning When Set** 5 MVS/XA 6 VS2 7 VS1
5(5)	SMF80RTY	1	binary	-	Record type (80 decimal)
6(6)	SMF80TME	4	binary	SVC 83	Time, in hundredths of a second, record was moved to SMF buffer
10(A)	SMF80DTE	4	packed	SVC 83	Date record was moved to SMF buffer, in form 00YYDDDF, where F is the sign
14(E)	SMF80SID	4	EBCDIC	SVC 83	System identification
18(12)	SMF80DES	2	binary	-	Descriptor flags: **Bit Meaning When Set** 0 The event is a violation 1 User is not defined to RACF 2 Record contains a version indicator (see SMF8OVER) 3 The event is a warning
20(14)	SMF80EVT	1	binary	-	Event code (see Table 1)
21(15)	SMF80EVQ	1	binary	-	Event code qualifier (see Table 1)
22(16)	SMF80USR	8	EBCDIC	ACEE (TIOT)	Identifier of the user for which this event is recorded (jobname is used if the user is not defined to RACF)
30(1E)	SMF80GRP	8	EBCDIC	ACEE (TIOT)	Group to which the user was connected (stepname is used if the user is not defined to RACF)
38(26)	SMF80REL	2	binary	-	Offset from beginning of the record header to the first relocate section
40(28)	SMF80CNT	2	binary	-	Count of the number of relocate sections
42(2A)	SMF80ATH	1	binary	-	Authorities used for executing commands or accessing resources: (see Note 1) **Bit Meaning When Set** 0 Normal authority check (resource access) 1 SPECIAL attribute (command execution) 2 OPERATIONS attribute (resource access) 3 AUDITOR attribute (command execution) 4 Installation exit processing (resource access) 5 Failsoft processing (resource access)
43(2B)	SMF80REA	1	binary	-	Reason for logging: (see Note 2) **Bit Meaning When Set** 0 Changes to this class of profile are being logged 1 User being audited 2 SPECIAL users being audited 3 Access to the resource is being audited due to the AUDIT option, a logging request from the RACHECK exit routine, or because the operator granted access during failsoft processing. 4 RACINIT failure 5 This command is always audited 6 Violation detected in command and CMDVIOL is in effect 7 Access to entity being audited due to GLOBALAUDIT option

Offset Dec(Hex)	Field Name	Length	Format	Source	Description
44(2C)	SMF80TLV	1	binary	-	Terminal level number of foreground user (zero if not available or not verified)
45(2D)	SMF80ERR	1	binary	-	Command processing error flag: (see Note 3) **Bit** **Meaning When Set** 0 Command had error, and could not back out some changes 1 No profile updates were made because of error in RACF processing
46(2E)	SMF80TRM	8	EBCDIC	-	Terminal ID of foreground user (zero If not available)
54(36)	SMF80JBN	8	EBCDIC	JMRJOB	Job name
62(3E)	SMF80RST	4	binary	JMRENTRY	Time (in hundredths of a second) that the reader recognized the JOB statement for this job
66(42)	SMF80RSD	4	packed	JMREDATE	Date the reader recognized the JOB statement for this job
70(46)	SMF80UID	8	EBCDIC	JMRUSEID	User identification field from the SMF common exit parameter area
78(4E)	SMF80VER	1	binary	RCVTVERS	Version indicator 7 = Version 1, Release 7
Relocate Section:					
0(0)	SMF80DTP	1	binary	-	Data type: (see Table 2)
1(1)	SMF80DLN	1	binary	-	Length of data that follows
2(2)	SMF80DTA	1-255	mixed	-	Data: (see Table 2)

Note 1: These flags indicate the authority checks made for the user who requested the action. The RACF commands use bits 0, 1, and 3; the RACF SVCs use bits 0, 2, and 4.

- Bit 0 indicates that the user's authority to issue the command or SVC was determined by the checks for a user with the SPECIAL, OPERATIONS, or AUDITOR attribute. This bit indicates that the tests were made, not that the user passed the tests and has authority to issue the command. This bit is not set on if the user has the AUDITOR attribute and entered the command with only those operands that require the AUDITOR attribute.

- Bit 1 indicates that the user has the SPECIAL attribute and used this authority to issue the command. If the user also has the AUDITOR attribute and entered the command with only those operands that require the AUDITOR attribute, this bit is not set on because the user did not use his authority as a user with the SPECIAL attribute.

- Bit 2 is set by the RACHECK and RACDEF SVCs and indicates that the user has the OPERATIONS attribute and used this authority to obtain access to the resource.

- Bit 3 indicates that the user has the AUDITOR attribute and used this authority to issue the command with operands that require the AUDITOR attribute.

- Bit 4 indicates that the user has authority because the exit routine indicated that the request is to be accepted without any further authority checks.

- Bit 5 indicates that resource access was granted by the operator during failsoft processing.

Note 2: These flags indicate the reason RACF produced the SMF record.

- Bit 0 is set when there are changes made to a profile in a class specified in the AUDIT operand of the SETROPTS command.

- Bit 1 is set when a user with the AUDITOR attribute specifies the UAUDIT operand on the ALTUSER command for a user and the user has changed RACF profiles with a RACF command, or a RACHECK or RACDEF SVC has been issued for the user.

- Bit 2 is set when a user with the AUDITOR attribute specifies the SAUDIT operand on the SETROPTS command and a user with the SPECIAL attribute has changed RACF profiles with a RACF command. However, if a user has both the SPECIAL and AUDITOR attributes and issues a command with operands that require only the AUDITOR attribute, RACF does not log this activity because SPECIAL authority was not used.

- Bit 3 is set if:

  - The AUDIT option in the resource profile specifies that attempts to access the resource be logged.

  - The RACHECK exit routine specifies unconditional logging.

  - The console operator grants the resource access during failsoft processing.

- Bit 4 is set when the RACINIT SVC fails to verify a user because of an invalid group, password, terminal, or OIDCARD.

- Bit 5 is set if the RVARY or SETROPTS command produced the SMF record. (The execution of these two commands always produce an SMF record.)

- Bit 6 is set when a user with the AUDITOR attribute specifies logging of command violations (with the CMDVIOL operand on the SETROPTS command) and RACF detects a violation.

- Bit 7 is set when attempts to access a RACF-protected resource are being logged, as requested by the GLOBALAUDIT option in the resource profile.

Note 3: These flags indicate errors during command processing and the extent of the processing.

- Bit 0 indicates that an error occurred that prevented the command from completing all updates requested, and the command was unable to back out the updates already done. If this bit is on, there may be an inconsistency between the profiles on the RACF data set, or between the profile for a data set and the RACF-indicator for the data set in the DSCB or catalog. The latter is also indicated by a bit in the command-related information for the ADDSD, ALTDSD, and DELDSD commands. For some commands (for example, ADDUSER), the inconsistency means an incompletely defined resource. For other commands, where the profiles are already defined (for example, ALTUSER), the inconsistency means that all changes were not made, but the profiles are still usable. (See Chapter 4, "Recovery Procedures" for additional information.)

This bit indicates a terminating error and should not be confused with a keyword violation or processing error where the command continues processing other operands.

- Bit 1 indicates that none of the requested changes were made, because either (1) a terminating error occurred before the changes were made, or (2) the command was able to back out the changes after a terminating error.

Table 1 (Event Codes and Event Code Qualifiers)

This table describes the event code (SMF80EVT) and event code qualifier (SMF80EVQ) fields.

Note that the event code qualifier is 0 if the recorded event is not a violation.

For event codes 8 through 25, an event code qualifier of 1 indicates one of the following:

- The command user is not RACF-defined.
- The command user is not authorized to change the requested profiles on the RACF data set.
- The command user does not have sufficient authority for any of the operands on the command.

For event codes 8 through 25, an event code qualifier of 2 indicates that the command user does not have sufficient authority to specify some of the operands, but RACF performed the processing for the operands for which the user has sufficient authority.

Event Code	Event Code Qual.	Description	Maximum SMF80CNT Value	Possible SMF80DTP Values
1		JOB INITIATION/TSO LOGON	1	20
	0	Successful RACINIT ENVIR=CREATE		
	1	Invalid password		
	2	Invalid group		
	3	Invalid OIDCARD		
	4	Invalid terminal		
	5	Invalid application		
	6	Revoked user attempting access		
	7	Userid has been automatically revoked		
	8	Successful RACINIT ENVIR=DELETE		
2		RESOURCE ACCESS	(see Note 3)	1,3,4,5,15,16,17,20,33,38 (see Notes 1 and 2)
	1	Insufficient authority		
	2	Resource not found		
	3	Warning message issued (WARNING option)		
3		END OF VOLUME (to add a new volume)	(see Note 3)	1,4,5,15,16,17,33,38 (see Note 2)
	1	Insufficient authority (DATASET only)		
4		RENAME DATA SET	(see Note 3)	1,2,5,15,17,33,38
	1	Invalid group (not defined)		
	2	No group (user not connected to group)		
	3	Insufficient authority		
	4	Rename error (data set already defined)		
	5	User not RACF-defined		
	6	Profile not found and PROTECT ALL(FAIL) is in effect		
	7	Profile not found and PROTECT ALL(WARNING) is in effect		
5		SCRATCH DATA SET OR TAPE VOLUME	(see Note 3)	1,5,15,17,33,38
	1	Resource not found		
	2	Invalid volume identification (DATASET only)		
6		DELETE 1 VOLUME OF MULTIVOLUME DATA SET OR TAPE VOLUME SET	5	1,5,15,17,38

	Event		Maximum	Possible
Event	Code		SMF80CNT	SMF80DTP
Code	Qual.	Description	Value	Values
7		DEFINE DATA SET OR TAPE VOLUME	(see Note 3)	1,5,15,17,18,19,33,38
	1	Invalid group (not defined) (DATASET only)		
	2	No group (user not connected to group)		
	3	Insufficient authority (User does not have at least CREATE authority) (DATASET only)		
	4	Define error (resource already defined)		
	5	User not RACF-defined		
	6	Profile not found and PROTECT ALL(FAIL) in effect		
	7	Profile not found and PROTECT ALL(WARNING) in effect		
8		ADDSD COMMAND336,10,38		
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
9		ADDGROUP COMMAND	3	6,37,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
10		ADDUSER COMMAND	(see Note 3)	6,7,8,28,37,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
11		ALTDSD COMMAND	4	6,10,11,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
12		ALTGROUP COMMAND	3	6,37,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
13		ALTUSER COMMAND	(see Note 3)	6,7,8,28,37,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
14		CONNECT COMMAND	2	6,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
15		DELDSD COMMAND	2	6,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		

Event Code	Event Code Qual.	Description	Maximum SMF80CNT Value	Possible SMF80DTP Values
16		DELGROUP COMMAND	2	6,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
17		DELUSER COMMAND	2	6,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
18		PASSWORD COMMAND	2	6,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
19		PERMIT COMMAND	(see note 3)	6,9,12,13,14,17,26,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
20		RALTER COMMAND	10	6,7,9,10,11,17,24,25,29,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
21		RDEFINE COMMAND	7	6,7,9,17,24,29,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
22		RDELETE COMMAND	4	6,9,17,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
23		REMOVE COMMAND	3	6,17,38
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
24		SETROPTS COMMAND	(see Note 3)	6,21,22,23,27,32,34,35,36
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		
25		RVARY COMMAND	4	6,27,30,31
	1	Insufficient authority (no update to RACF data set)		
	2	Insufficient authority (partial update to RACF data set)		

Note 1: The SMF80DTP value 4 (access authority allowed) can be less than the SMF80DTP value 3 (access authority requested) in two cases:

- When RACF authorizes access to a user who requested access to a data set because the user has the OPERATIONS attribute.
- When the RACHECK exit routine returns a return code of 12, which indicates that the request should be granted.

Note 2: The SMF80DTP value of 16 appears only when RACHECK received an old volume (OLDVOL) as input. The value of 33 appears when a generic profile is used. In cases where RACDEF is used to rename a resource (SMF80EVT=4), the data type 33 relocate section can hold a generic resource name that is either the old or the new name, or it can hold the generic profile that protects the old or the new name.

Note 3: The number of relocate sections varies, depending on the class and options used as well as the number of entries in the class descriptor table (CDT).

Table 2 (Relocate Section Variable Data)

This table describes the variable data elements of the relocate section.

Data Type	**Data Length**	**Format**	**Description**
(SMF80DTP)	**(SMF80DLN)**		**(SMF80DTA)**
1	1-44	EBCDIC	Old data set name (RACHECK, RACDEF)
2	1-44	EBCDIC	New data set name (RACDEF)
3	1	binary	Access authority requested (RACHECK) (see Note 1)
4	1	binary	Access authority allowed (RACHECK, RACDEF) (see Note 1)
5	1	binary	Data set level number (00-99)
6	1-255	mixed	RACF command-related data (see Table 3)
7	1-255	EBCDIC	DATA installation-defined data (ADDUSER, ALTUSER, RALTER, RDEFINE)
8	1-20	EBCDIC	NAME user-name (ADDUSER, ALTUSER)
9	1-44	EBCDIC	Resource name (PERMIT, RALTER, RDEFINE, RDELETE)
10	7	EBCDIC	Volume serial (ALTDSD ADDVOL, RALTER ADDVOL, ADDSD VOLUME). When set on, bit 0 of the first byte indicates that the volume was not processed. Bytes 2-7 contain the volume serial number.
11	7	EBCDIC	Volume serial (ALTDSD DELVOL, RALTER DELVOL). When set on, bit 0 of the first byte indicates that the volume was not processed. Bytes 2-7 contain the volume serial.
12	9-243		1 to 27 ID names (PERMIT), each organized as follows:
	1	binary	Processing flags: **Bit Meaning When Set** 0 ID ignored because of processing error (see Note 2) 1-7 Reserved
	8	EBCDIC	ID name
13	1-44	EBCDIC	FROM resource name (PERMIT)
14	12	EBCDIC	VOLUME volume serial (6 bytes) followed by FVOLUME volume serial (6 bytes) (PERMIT)
15	6	EBCDIC	VOLSER volume serial (RACDEF, RACHECK) (Note that when RACHECK receives a DATASET profile as input, the volume serial logged is the first volume serial contained in the profile's list of volume serials.)
16	6	EBCDIC	OLDVOL volume serial (RACDEF, RACHECK) (Note that when RACHECK receives a DATASET profile as input, the volume serial logged is the first volume serial contained in the profile's list of volume serials.)
17	1-8	EBCDIC	Class name (RACDEF, RACHECK, RDEFINE, RALTER, RDELETE, PERMIT)
18	1-44	EBCDIC	MENTITY model resource name (RACDEF)
19	6	EBCDIC	Volume serial of model resource (RACDEF)
20	8	EBCDIC	Application name (RACHECK, RACINIT processed)
21	9	EBCDIC	Current class options (set by SETROPTS or RACF initialization) Byte 1: **Bit Meaning When Set** 0 Statistics are in effect 1 Auditing is in effect 2 Protection is in effect 3 Generic profile processing is in effect 4 Generic command processing is in effect 5-7 Reserved Bytes 2-9: Class name
22	8	EBCDIC	Class name from STATISTICS/NOSTATISTICS keyword (SETROPTS)
23	8	EBCDIC	Class name from AUDIT/NOAUDIT keyword (SETROPTS)
24	2-40	EBCDIC	Resource name from ADDMEM keyword (RDEFINE, RALTER) Byte 1: **Bit Meaning When Set** 0 Resource name not processed 1 Resource name ignored because command user lacked sufficient authority to perform the operation Bytes 2-40: Resource name

Data Type (SMF80DTP)	Data Length (SMF80DLN)	Format	Description (SMF80DTA)
25	2-40	EBCDIC	Resource name from DELMEM keyword (RALTER). Bit 0 of the first byte, when set on, indicates that the resource name was not processed. Bytes 2-40 contain the resource name.
26	8	EBCDIC	Class name from FCLASS keyword (PERMIT)
27	8	EBCDIC	Class name from CLASSACT/NOCLASSACT keyword (SETROPTS, RVARY)
28	9	mixed	Class name from CLAUTH/NOCLAUTH keyword (ADDUSER, ALTUSER). Bit 1 of the first byte, when set on, indicates that the class was ignored because the command user did not have sufficient authority to perform the operation. Bytes 2-9 contain the class name.
29	1-255	EBCDIC	Application data (RDEFINE, RALTER)
30	12-55	mixed	RACF data set status (RVARY, RACF initialization) Byte 1: **Bit** **Meaning When Set** 0 Data set is active 1 Data set is backup 2-7 Reserved Bytes 2-4 Unit name Bytes 5-10 Volume Byte 11: Sequence number Byte 12: 1-44 character data set name
31	1-44	EBCDIC	Data set name from DATASET operand (RVARY)
32	89	mixed	**Byte** **Description** 1 Password interval value 2 Password history value 3 Userid revoke value 4 Password warning level value 5-84 Password syntax rules value 85 Userid inactive interval 86-89 Indicators **Bit** **Meaning** 0 MODEL(GDG) in effect 1 MODEL(USER) in effect 2 MODEL(GROUP) in effect 3 GRPLIST in effect 4-31 Reserved
33	2-45	mixed	Flag bits **Byte** **Bit** **Description** 1 0 1 = Resource name is generic; 0 = Generic profile is used 1 1 = The **old** name of a RACDEF-renamed data set 0 = The **new** name of a RACDEF-renamed data set 2-7 Reserved 2-45 Generic resource name or name of generic profile used
34	8	EBCDIC	Class name from GENERIC/NOGENERIC (SETROPTS)
35	8	EBCDIC	Class name from GENCMD/NOGENCMD (SETROPTS)
36	8	EBCDIC	Class name from GLOBAL/NOGLOBAL (SETROPTS)
37	1-44	EBCDIC	Model name
38	8	EBCDIC	Userid or group name that owns the profile (RACHECK, RACDEF, and all the RACF commands that produce log records, except SETROPTS and RVARY). During DEFINE operations, this field contains the owner that the profile is defined with; in all other operations, it contains the current owner. Thus, for owner changes, it contains the old owner.
39	9-243		1 to 27 program names (PERMIT), each organized as follows:
	1	binary	Processing flags: **Bit** **Meaning When Set** 0 program ignored because of processing error (see Note 2) 1-7 Reserved
	8	EBCDIC	program name

40	1	binary	Category name (ADDSD, ALTDSD, ADDUSER, ALTUSER, RDEFINE, RALTER commands, and RACDEF SVC) to be added to the profile
	1	binary	Processing flags: **Bit** **Meaning When Set** 0 category name ignored because of processing error 1-7 Reserved
	8	EBCDIC	Category name added
41	1	binary	Category name (ALTDSD, ALTUSER, and RALTER commands) to be deleted from the profile
	1	binary	Processing flags: **Bit** **Meaning When Set** 0 category name ignored because of processing error 1-7 Reserved
	8	EBCDIC	Category name deleted

Note 1: The access flags are:

Bit	Access Authority
0	ALTER
1	CONTROL
2	UPDATE
3	READ
4	NONE

Note 2: This bit is turned on for each ID in the list (data type 12) and each program name in the list (data type 39) that was not processed because of a nonterminating error, such as userids (specified on the ID operand of the PERMIT command) that are not defined to RACF. If a terminating error, such as a RACF manager error, occurred while processing an ID, this bit is turned on for all remaining IDs that were not processed.

Table 3 (Data Type 6 Command-Related Data)

- ADDSD
- ADDUSER
- ALTUSER
- ALTDSD
- CONNECT
- PERMIT
- RALTER
- RDEFINE
- RVARY
- SETROPTS

This table describes the RACF command-related data associated with data type 6. The actual format and content of the data depends upon the command being logged. Command-related data will not appear in the SMF record if the command user is not RACF-defined. Some of the commands also omit the command-related data if the user is not authorized for the requested profile on the RACF data set.

The table is arranged by event code. In each description, the keyword flags contain one flag for each possible keyword that you can specify (explicitly or by default) on the command. The 'flags for keywords specified' field indicates whether the keyword was specified or defaulted.

The 'flags for keywords ignored because of insufficient authority' indicates whether the keyword was ignored because the user did not have sufficient authority to use the keyword. The event code qualifier (SMF80EVQ), described in Table 1, is set to 1 if the command user does not have sufficient authority for any of the keywords specified or taken as defaults. The event code qualifier is set to 2 if the command user does not have sufficient authority for some (but not all) of the keywords specified or taken as defaults. In the latter case, the command continues processing the authorized operands.

The 'flags for keywords ignored due to error conditions' field indicate individual keywords that were not processed for reasons other than insufficient authority. Not all commands (event codes 8-25) have these flags. The keyword errors are not terminating errors (like the errors indicated in SMF80ERR) and the command continues processing other specified operands. In the event of a terminating error, these flags do not necessarily indicate what processing was done or not done. Any keyword errors occurring before the terminating error are indicated, but the keywords not processed because of a terminating error are not indicated. The bits in SMF80ERR indicate whether or not RACF already made changes to the RACF data set before the terminating error and if it backed out the changes successfully.

Other fields in the command-related data field indicate the subfields specified (or defaulted) for keywords. The fields are flags for subfields that are keywords (such as SUCCESS subfield of AUDIT); they are data for subfields such as owner name or group name.

For example, if the owner of the profile for USERA issues the command:

```
ALTUSER USERA ADSP GRPACC SPECIAL OWNER(USERB)
```

and USERB, the requested new owner is not RACF-defined, then the command-related data would appear in the log record as:

```
012C0000 00040000 00080000 00E4E2C5
D9C14040 40000000 00000000 00000000
00000000 000000E4 E2C5D9C2 40404000
00000000
```

The first word indicates the keywords specified. The second word indicates the user does not have sufficient authority to use the SPECIAL keyword. The third word indicates there was an error processing the OWNER keyword. At offset X'0D' is the name of the user profile being altered. At offset X'27' is the name of the owner specified on the command. RACF processed the ADSP and GRPACC keywords.

Note: If you use SMF records to reconstruct a RACF data set, passwords and OIDCARDs are not contained in the records and require special handling, and statistics updates are not recorded.

Event Code	Data Length	Format	Description
8			**ADDSD command**
	2	binary	Flags for keywords specified: **Byte Bit Keyword specified** 0 0 VOLUME 1 UNIT 2 UACC 3 OWNER 4 AUDIT 5 SET 6 NOSET 7 LEVEL 1 0 PASSWORD 1 DATA 2 MODEL 3 WARNING 4 GENERIC 5 SECLEVEL 6 ADDCATEGORY 7 NOTIFY
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	44	EBCDIC	Data set name
	8	EBCDIC	Type (UNIT keyword)
	1	binary	Flags for UACC keyword: **Bit Authority specified** 0 ALTER 1 CONTROL 2 UPDATE 3 READ 4-6 Reserved 7 NONE

Event Code	Data Length	Format	Description
	8	EBCDIC	Userid or group name (OWNER Keyword)
	1	binary	Flags for AUDIT keyword: **Bit Option specified** 0 ALL 1 SUCCESS 2 FAILURES 3 NONE 4-5 SUCCESS qualifier codes 6-7 FAILURES qualifier codes
	1	binary	nn (LEVEL Keyword)
	1	binary	Flags for RACF processing: **Bit Meaning** 0 Data set profile inconsistent with RACF indicator. 1 Generic profile name specified 2-6 Reserved 7 User to be notified when this profile denies access
	2	binary	Flags for keywords specified **Byte Bit Keyword specified** 0 0 SETONLY 1 TAPE 2 FILESEQ 3 RETPD 4 ERASE 5 FROM 6 FCLASS 7 FVOLUME 1 0 FGENERIC 1-7 Reserved
	2	binary	Flags for keywords ignored. Same format as flags for keywords specified.
	4		Reserved
	15	EBCDIC	File sequence number
	15	EBCDIC	Retention period
	8	EBCDIC	FROM class name
	44	EBCDIC	FROM resource name
	8	EBCDIC	FROM volume serial
	44	EBCDIC	SECLEVEL name
9			**ADDGROUP command**
	1	binary	Flags for keywords specified: **Bit Keyword specified** 0 SUPGROUP 1 OWNER 2 NOTERMUACC 3 TERMUACC 4 DATA 5 MODEL 6-7 Reserved
	1	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	8	EBCDIC	Group name
	8	EBCDIC	Superior group name (SUPGROUP keyword)
	8	EBCDIC	Userid or group name (OWNER keyword)

Event Code	Data Length	Format	Description
10			**ADDUSER command**
			* The data for event code 10 is identical to the data for event code 13, with these exceptions.
	4	binary	Flags for keywords specified: **Byte Bit Keyword specified** 0 0 DFLTGRP 1 Reserved 2 PASSWORD 3 NOPASSWORD 4 NAME 5 AUTHORITY 6 DATA 7 GRPACC 1 0 NOGRPACC 1 UACC 2 ADSP 3 NOADSP 4 OWNER 5 SPECIAL 6 NOSPECIAL 7 OPERATIONS 2 0 NOOPERATIONS 1 CLAUTH 2 NOCLAUTH 3 AUDITOR 4 NOAUDITOR 5 OIDCARD 6 NOOIDCARD 7 REVOKE 3 0 RESUME 1 UAUDIT 2 NOUAUDIT 3 MODEL 4 NOMODEL 5 WHEN 6 ADDCATEGORY 7 DELCATEGORY
	4	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	4	binary	*Reserved
	1	binary	Flags for other violations: **Bit Violation** *0 Command invoker does not have CLAUTH attribute of USER 1 Command invoker does not have sufficient authority to group *2-7 Reserved
	8	EBCDIC	Userid
	8	EBCDIC	Group-name (DFLTGRP keyword)
	8	EBCDIC	Group id
	1	binary	Flags for AUTHORITY keyword: **Bit Authority specified** 0 JOIN 1 CONNECT 2 CREATE 3 USE 4-7 Reserved
	1	binary	Flags for UACC keyword: **Bit Authority specified** 0 ALTER 1 CONTROL 2 UPDATE 3 READ 4-6 Reserved 7 NONE
	8	EBCDIC	Userid or group name (OWNER keyword)

Event Code	Data Length	Format	Description
	2	binary	Flags for classes specified (CLAUTH keyword)
			Byte **Bit** **Option specified**
			0 0-1 Reserved
			2 USER
			3 Reserved
			4 DASDVOL
			5 TAPEVOL
			6 TERMINAL
			7 Reserved
			1 0-7 Reserved
	2	binary	Flags for classes ignored because of insufficient authority: Same as for flags for classes specified. Note that if all classes specified are ignored because of insufficient authority, then the 'flags for keywords ignored because of insufficient authority' field indicates that CLAUTH was ignored.
	2	binary	Flags for SECLEVEL/NOSECLEVEL keyword specified
			Byte **Bit** **Option specified**
			0 0 SECLEVEL
			1 NOSECLEVEL
			2-7 Reserved
			1 0-7 Reserved
	2	binary	Flags for SECLEVEL/NOSECLEVEL keyword ignored: Same as for SECLEVEL/NOSECLEVEL specified flags.
	2	binary	Flags for classes ignored because of processing error:
			Byte **Bit** **Option specified**
			0 0 SECLEVEL
			1 NOSECLEVEL
			2-7 Reserved
			1 0-7 Reserved
	3	EBCDIC	Logon time (packed); if time is not specified, this field contains binary zeroes.
	3	EBCDIC	Logoff time (packed); if time is not specified, this field contains binary zeroes.
	1	binary	Logon day
			Bit **Day specified**
			0 Sunday
			1 Monday
			2 Tuesday
			3 Wednesday
			4 Thursday
			5 Friday
			6 Saturday
			7 Day not specified
	4	EBCDIC	REVOKE date
	4	EBCDIC	RESUME date
	44	EBCDIC	SECLEVEL name specified
11			**ALTDSD command**
	2	binary	Flags for keywords specified:
			Byte **Bit** **Keyword specified**
			0 0 OWNER
			1 UACC
			2 AUDIT
			3 LEVEL
			4 ADDVOL
			5 DELVOL
			6 SET
			7 NOSET
			1 0 GLOBALAUDIT
			1 VOLUME
			2 password
			3 UNIT
			4 ALTVOL
			5 DATA
			6-7 Reserved
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	2	binary	Flags for keywords ignored because of error conditions: Same format as flags for keywords specified.

Event Code	Data Length	Format	Description		
	44	EBCDIC	Data set name		
	8	EBCDIC	Userid or group name (OWNER keyword)		
	1	binary	Flags for UACC keyword:		
			Bit	**Authority specified**	
			0	ALTER	
			1	CONTROL	
			2	UPDATE	
			3	READ	
			4-6	Reserved	
			7	NONE	
	1	binary	Flags for AUDIT keyword:		
			Bit	**Option specified**	
			0	ALL	
			1	SUCCESS	
			2	FAILURES	
			3	NONE	
			4-5	SUCCESS qualifier codes	
			6-7	FAILURES qualifier codes	
	1	binary	nn (LEVEL keyword)		
	1	binary	Flags for GLOBALAUDIT keyword: Same format as flags for AUDIT keyword.		
	6	EBCDIC	Volume serial ID (VOLUME keyword)		
	8	EBCDIC	Unit information		
				1	binary Flags for RACF processing:
			Bit	**Meaning**	
			0	Profile inconsistent with RACF indicator.	
			1	Generic profile name specified	
			2-7	Reserved	
	2	binary	Additional keywords specified:		
			Byte	**Bit**	**Keyword specified**
			0	0	GENERIC
				1	WARNING
				2	NOWARNING
				3	ERASE
				4	NOERASE
				5	RETPD
				6	NOTIFY
				7	NONOTIFY
			1	0	SECLEVEL
				1	ADDCATEGORY
				2	DELCATEGORY
				3	NOSECLEVEL
				4-7	Reserved
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.		
	2	binary	Flags for keywords ignored because of a processing error: Same format as flags for keywords specified.		
	2	EBCDIC	Retention period		
	8	EBCDIC	User to be notified when access denied.		
	44	EBCDIC	SECLEVEL name		

Event Code	Data Length	Format	Description
12			**ALTGROUP command**
	1	binary	Flags for keywords specified: **Bit Keyword specified** 0 SUPGROUP 1 OWNER 2 NOTERMUACC 3 TERMUACC 4 DATA 5 MODEL 6-7 Reserved
	1	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	1	binary	Flags for other violations: **Bit Violation** 0 Lack of proper authority to old SUPGROUP 1-7 Reserved
	8	EBCDIC	Group name
	8	EBCDIC	Superior group name (SUPGROUP keyword)
	8	EBCDIC	Userid or group name (OWNER keyword)
	1	binary	Flags for keywords ignored because of error conditions: Same format as flags for keywords specified.
13			**ALTUSER command** * The data for event code 13 is identical to the data for event code 10, with these exceptions.
	4	binary	Flags for keywords specified:

Byte	Bit	Keyword specified
0	0	DFLTGRP
	1	GROUP
	2	PASSWORD
	3	NOPASSWORD
	4	NAME
	5	AUTHORITY
	6	DATA
	7	GRPACC
1	0	NOGRPACC
	1	UACC
	2	ADSP
	3	NOADSP
	4	OWNER
	5	SPECIAL
	6	NOSPECIAL
	7	OPERATIONS
2	0	NOOPERATIONS
	1	CLAUTH
	2	NOCLAUTH
	3	AUDITOR
	4	NOAUDITOR
	5	OIDCARD
	6	NOOIDCARD
	*7	REVOKE
3	*0	RESUME
	*1	UADIT
	*2	NOUAUDIT
	3	MODEL
	4	NOMODEL
	5-7	Reserved

Event Code	Data Length	Format	Description
	4	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	4	binary	Flags for keywords ignored because of error conditions: Same format as flags for keywords specified.
	1	binary	Flags for other violations:
			Bit **Violation**
			*0 Reserved
			1 Command invoker does not have sufficient authority to group
			*2 Command invoker does not have sufficient authority to user
			3-7 Reserved
	8	EBCDIC	Userid
	8	EBCDIC	Group name (DFLTGRP keyword)
	8	EBCDIC	*Group name (GROUP keyword)
	1	binary	Flags for AUTHORITY keyword:
			Bit **Authority specified**
			0 JOIN
			1 CONNECT
			2 CREATE
			3 USE
			4-7 Reserved
	1	binary	Flags for UACC keyword:
			Bit **Authority specified**
			0 ALTER
			1 CONTROL
			2 UPDATE
			3 READ
			4-6 Reserved
			7 NONE
	8	EBCDIC	Userid (OWNER keyword)
	2	binary	Flags for classes specified (CLAUTH/NOCLAUTH keywords)
			Byte **Bit** **Option specified**
			0 0-1 Reserved
			2 USER
			3 Reserved
			4 DASDVOL
			5 TAPEVOL
			6 TERMINAL
			7 Reserved
			1 0-7 Reserved
	2	binary	Flags for classes ignored because of insufficient authority: Same format as flags for classes specified. Note that if all classes specified are ignored because of insufficient authority, then the 'flags for keywords ignored because of insufficient authority' field indicates that CLAUTH or NOCLAUTH was ignored.

Event Code	Data Length	Format	Description
14			**CONNECT command**
	2	binary	Flags for keywords specified: **Byte Bit Keyword specified** 0 0 GROUP 1 UACC 2 AUTHORITY 3 ADSP 4 NOADSP 5 REVOKE 6 RESUME 7 GRPACC 1 0 NOGRPACC 1 OPERATIONS 2 NOOPERATIONS 3 SPECIAL 4 NOSPECIAL 5 AUDITOR 6 NOAUDITOR 7 OWNER
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	8	EBCDIC	Userid
	8	EBCDIC	Group name (GROUP keyword)
	1	binary	Flags for UACC keyword: **Bit Authority specified** 0 ALTER 1 CONTROL 2 UPDATE 4-6 Reserved 7 NONE
	1	binary	Flags for AUTHORITY keyword: **Bit Authority specified** 0 JOIN 1 CONNECT 2 CREATE 3 USE 4-7 Reserved
	2	EBCDIC	Reserved
	8	EBCDIC	Userid or group name (OWNER keyword)
	8	EBCDIC	Reserved
15			**DELDSD command**
	1	binary	Flags for keywords specified or taken as defaults: **Bit Keyword specified** 0 SET 1 NOSET 2 VOLUME 3 GENERIC 4-7 Reserved
	1	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	44	EBCDIC	Data set name
	6	EBCDIC	Volume serial ID (VOLUME keyword)
	1	binary	Flags for RACF processing: **Bit Meaning** 0 Profile inconsistent with RACF indicator 1 Generic profile name specified 2-7 Reserved

Event Code	Data Length	Format	Description		
16			**DELGROUP command**		
	8	EBCDIC	Group name		
17			**DELUSER command**		
	8	EBCDIC	Userid		
18			**PASSWORD command**		
	1	binary	Flags for keywords specified:		
			Bit	**Keyword specified**	
			0	INTERVAL	
			1	USER	
			2	PASSWORD	
			3-7	Reserved	
	1	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.		
	1	binary	Flags for keywords ignored because of error conditions: Same format as flags for keywords specified.		
	4	binary	Change-interval (INTERVAL keyword)		
	8	EBCDIC	Userid (USER keyword)		
19			**PERMIT command**		
	2	binary	Flags for keywords specified or taken as defaults:		
			Byte	**Bit**	**Keyword specified**
			0	0	CLASS
				1	ID
				2	ACCESS
				3	FROM
				4	DELETE
				5	FCLASS
				6	VOLUME
				7	FVOLUME
			1	0	GENERIC
				1	FGENERIC
				2	RESET
				3-7	Reserved
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.		
	2	binary	Flags for keywords ignored because of error conditions: Same format as flags for keywords specified.		
	2	binary	Flags for CLASS keyword:		
			Byte	**Bit**	**Option specified**
			0	0-2	Reserved
				3	DATASET
				4	DASDVOL
				5	TAPEVOL
				6	TERMINAL
				7	Reserved
			1	0-7	Reserved
	1	binary	Flags for ACCESS keyword:		
			Bit	**Authority specified**	
			0	ALTER	
			1	CONTROL	
			2	UPDATE	
			3	READ	
			4-6	Reserved	
			7	NONE	
	2	binary	Flags for FCLASS keyword: Same format as flags for CLASS keyword.		

Event Code	Data Length	Format	Description
20			**RALTER command**
			* The data for event code 20 is identical to the data for event code 21, with these exceptions.
	2	binary	Flags for keywords specified: **Byte Bit Keyword specified** 0 0 DATA 1 OWNER 2 UACC 3 LEVEL 4 AUDIT *5 GLOBALAUDIT *6 ADDVOL *7 DELVOL 1 0 ADDMEM 1 DELMEM 2 APPLDATA 3-4 Reserved 5 WARNING 6 NOWARNING 7 Reserved
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	2	binary	Flags for classname: **Byte Bit Option specified** 0 0-3 Reserved 4 DASDVOL 5 TAPEVOL 6 TERMINAL 7 Reserved 1 0 Generic resource name specified 1-7 Reserved
	8	EBCDIC	Userid or group name (OWNER keyword)
	1	binary	Flags for UACC keyword: **Bit Authority specified** 0 ALTER 1 CONTROL 2 UPDATE 3 READ 4-6 Reserved 7 NONE
	1	binary	nn (LEVEL keyword)
	1	binary	Flags for AUDIT keyword: **Bit Option specified** 0 ALL 1 SUCCESS 2 FAILURES 3 NONE 4-5 SUCCESS qualifier codes 6-7 FAILURES qualifier codes
	1	binary	*Flags for GLOBALAUDIT keyword: Same format as flags for AUDIT keyword.

Event Code	Data Length	Format	Description
21			**RDEFINE command**
			* The data for event code 21 is identical to the data for event code 20, with these exceptions.
	2	binary	Flags for keywords specified:
			Byte Bit Keyword specified
			0 0 DATA
			1 OWNER
			2 UACC
			3 LEVEL
			4 AUDIT
			5-7 *Reserved
			1 0 ADDMEM
			1 DELMEM
			2 APPLDATA
			3-4 Reserved
			5 WARNING
			6 NOWARNING
			7 Reserved
	2	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.
	2	binary	Flags for class-name
			Byte Bit Option specified
			0 0-3 Reserved
			4 DASDVOL
			5 TAPEVOL
			6 TERMINAL
			7 Reserved
			1 0 Generic resource name specified
			1-7 Reserved
	8	EBCDIC	Userid or group name (OWNER keyword)
	1	binary	Flags for UACC keyword:
			Bit Authority specified
			0 ALTER
			1 CONTROL
			2 UPDATE
			3 READ
			4-6 Reserved
			7 NONE
	1	binary	nn (LEVEL keyword)
	1	binary	Flags for AUDIT keyword:
			Bit Option specified
			0 ALL
			1 SUCCESS
			2 FAILURES
			3 NONE
			4-5 SUCCESS qualifier codes
			6-7 FAILURES qualifier codes
	1	binary	*Reserved

* The data for event code 21 is identical to the data for event code 20, with these exceptions.

Event Code	Data Length	Format	Description		
22			**RDELETE command**		
	2	binary	Flags for class-name		
			Byte	**Bit**	**Option specified**
			0	0-3	Reserved
				4	DASDVOL
				5	TAPEVOL
				6	TERMINAL
				7	Reserved
			1	0	Generic resource name specified
				1-7	Reserved
23			**REMOVE command**		
	1	binary	Flags for keywords specified:		
			Bit	**Keyword specified**	
			0	GROUP	
			1	OWNER	
			2-7	Reserved	
	1	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.		
	8	EBCDIC	Userid (to be removed)		
	8	EBCDIC	Group name (GROUP keyword)		
	8	EBCDIC	Userid or group name (OWNER keyword)		
24			**SETROPTS command**		
	3	binary	Flags for keywords specified:		
			Byte	**Bit**	**Keyword specified**
			0	0	TAPE
				1	NOTAPE
				2	INITSTATS
				3	NOINITSTATS
				4	SAUDIT
				5	NOSAUDIT
				6	STATISTICS
				7	NOSTATISTICS
			1	0	AUDIT
				1	NOAUDIT
				2	TERMINAL
				3	NOTERMINAL
				4	INTERVAL(PASSWORD)
				5	CMDVIOL
				6	NOCMDVIOL
				7	DASD
			2	0	NODASD
				1	CLASSACT
				2	NOCLASSACT
				3	HISTORY or NOHISTORY
				4	WARNING or NOWARNING
				5	REVOKE or NOREVOKE
				6	NORULES or RULEn
				7	INACTIVE INTERVAL
	3	binary	Flags for keywords ignored because of insufficient authority: Same format as flags for keywords specified.		
	1	binary	Flags for STATISTICS or NOSTATISTICS keyword:		
			Byte	**Bit**	**Option specified**
			0	0-2	Reserved
				3	DATASET
				4	DASDVOL
				5	TAPEVOL
				6	TERMINAL
				7	Reserved

Event Code	Data Length	Format	Description
	1	binary	Flags for keywords ignored: **Bit** **Keyword Specified** 0 MODEL-GDG 1 MODEL-NOGDG 2 MODEL-USER 3 MODEL-NOUSER 4 MODEL-GROUP 5 MODEL=NOGROUP 6 GRPLIST 7 NOGRPLIST
	1	binary	Flags for AUDIT or NOAUDIT keyword: **Bit** **Option specified** 0 Reserved 1 GROUP 2 USER 3 DATASET 4 DASDVOL 5 TAPEVOL 6 TERMINAL 7 Reserved
	1	binary	Flags for keywords specified: **Bit** **Keyword Specified** 0 MODEL-GDG 1 MODEL-NOGDG 2 MODEL-USER 3 MODEL-NOUSER 4 MODEL-GROUP 5 MODEL-NOGROUP 6 GRPLIST 7 NOGRPLIST
	1	binary	Change-interval (INTERVAL keyword)
	1	binary	Flags for TERMINAL keyword: **Bit** **Option specified** 0-2 Reserved 3 READ 4-6 Reserved 7 NONE
	1	binary	Flags for current statistics options after SETROPTS has executed: **Bit** **Option specified** 0 Reserved 1 Bypass RACINIT statistics 2 Bypass data set statistics 3 Bypass tape volume statistics 4 Bypass DASD volume statistics 5 Bypass terminal statistics 6 Bypass ADSP attribute 7 Reserved
	1	binary	Flags for current audit options after SETROPTS has executed: **Bit** **Option specified** 0 Reserved 1 Log group class 2 Log user class 3 Log data set class 4 Log DASD volume class 5 Log tape volume class 6 Log terminal class 7 Reserved

Event Code	Data Length	Format	Description
	1	binary	Reserved
	2	binary	Flags for miscellaneous options after SETROPTS has executed:
			Byte **Bit** **Option specified**
			0 0 Perform terminal authorization checking
			1 Terminal UACC=NONE (if this bit is off, terminal UACC=READ)
			2 Log RACF command violations
			3 Log SPECIAL user activity
			4-7 Reserved
			1 0 Tape volume protection is in effect
			1 DASD volume protection is in effect
			2 Generic profile processing is in effect for the DATASET class
			3 Generic command (GENCDM) processing is in effect for the DATASET class
			4 REALDSN is in effect
			5 JES-XBMALLRACF is in effect
			6 JES-EARLYVERIFY is in effect
			7 JES-BATCHALLRACF in effect
	1	binary	Maximum password interval
	1	binary	Password history generation value
	1	binary	Password revoke value
	1	binary	Password warning level
	80	binary	Password syntax rules (eight rules). Each rule has following basic format:
			Byte **Description**
			0 Starting length value
			1 Ending length value
			2-9 Character content rules for each of the eight possible positions. The character values are:
			L = Alphanumeric
			A = Alphabetic
			N = Numeric
			V = Vowel
			C = Consonant
			W = No vowels
	1	binary	Userid inactive interval
	3	binary	Flags for keywords specified:
			Byte **Bit** **Keyword specified**
			0 0 ADSP
			1 NOADSP
			2 GENERIC
			3 NOGENERIC
			4 GENCMD
			5 NOGENCMD
			6 GLOBAL
			7 NOGLOBAL
			1 0 PREFIX
			1 NOPREFIX
			2 REALDSN
			3 NOREALDSN
			4 JES-XBMALLRACF
			5 JES-NOXBMALLRACF
			6 JES-BATCHALLRACF
			7 JES-NOBATCHALLRACF
			2 0 JES-EARLYVERIFY
			1 JES-NOEARLYVERIFY
			2 REFRESH
			3-7 Reserved
	3	binary	Flags for keywords specified but ignored because of insufficient authority: Same format as flags for keywords specified.
	8	char	Single-level data set name prefix
	16	char	Reserved

Event	Data			
Code	Length	Format	Description	
25			**RVARY command**	
	1	binary	Flags for keywords specified:	
			Bit	**Keyword specified**
			1	INACTIVE
			2	NOTAPE
			3	NOCLASSACT
			4-7	Reserved
	1	binary	Flags for other violations:	
			Bit	**Violation**
			0	Command denied by operator
			1	Nonzero code returned from RACF manager during ACTIVE processing
			2-7	Reserved

Record Type 81 - RACF Initialization Record

RACF (module ICHSEC00) writes record type 81 at the completion of the initialization of RACF. This record contains:

- The record type
- Time stamp (time and date)
- Processor identification
- Data set name of each RACF data set
- Volume identification of each RACF data set
- Unit name of the RACF data set
- Data set name of the UADS data set
- Volume identification of the UADS data set
- RACF options
- The maximum password interval

The format of record type 81 is:

Offset Dec(Hex)	Field Name	Length	Format	Source	Description
0(0)	SMF81LEN	2	binary	-	Total record length
2(2)	-	2	binary	-	Reserved
4(4)	SMF81FLG	1	binary	-	System indicator: **Bit** **Meaning When Set** 5 - MVS/XA 6 - VS2 7 - VS1
5(5)	SMF81RTY	1	binary	-	Record type (81 decimal)
6(6)	SMF81TME	4	binary	SVC 83	Time, in hundredths of a second, record was moved to SMF buffer
10(A)	SMF81DTE	4	packed	SVC 83	Date record was moved to SMF buffer, in the form 00YYDDDF, where F is the sign
14(E)	SMF81SID	4	EBCDIC	SVC 83	System identification
18(12)	SMF81RDS	44	EBCDIC	RCVTDSN	Data set name of the RACF data set for this IPL (all blanks if RACF is not active)
62(3E)	SMF81RVL	6	EBCDIC	RACFVOL	Volume identification of RACF data set (all blanks if RACF is not active)
68(44)	SMF81RUN	3	EBCDIC	UCBNAME	Unit name of RACF data set (all blanks if RACF is not active)
71(47)	SMF81UDS	44	EBCDIC	RCVTUADS	Data set name of the UADS data set for this IPL
115(73)	SMF81UVL	6	EBCDIC	RCVTUVOL	Volume identification of the UADS data set
121(79)	SMF81OPT	1	binary	RCVTSTAT	Options indicator: **Bit** **Meaning When Set** 0 No RACINIT statistics are recorded 1 No DATASET statistics are recorded 2 RACINIT preprocessing exit routine, ICHRIX01, is active 3 RACHECK preprocessing exit routine, ICHRCX01, is active 4 RACDEF preprocessing exit routine, ICHRDX01, is active 5 RACINIT postprocessing exit routine, ICHRIN02, is active 6 RACHECK postprocessing exit routine, ICHRCX02, is active 7 New password exit routine, ICHPWX01, is active
122(7A)	SMF81OP2	1	binary	RCVTSTAT	Options indicator: **Bit** **Meaning When Set** 0 No tape volume statistics are recorded 1 No DASD volume statistics are recorded 2 No terminal statistics are recorded 3 Command exit routine ICHCNX00 is active 4 Command exit routine ICHCCX00 is active 5 ADSP is not active 6 Encryption exit routine, ICHDEX01, is active 7 Naming convention table, ICHNCV00, is present

Offset Dec(Hex)	Field Name	Length	Format	Source	Description
Relocate Section:					
123(7B)	SMF81OP3	1	binary		Options indicator:
					Bit **Meaning When Set**
				RCVTSTA1	0 Tape volume protection is in effect
				RCVTFLGS	1 No duplicate data set names are to be defined
				RCVTSTA1	2 DASD volume protection is in effect
					3 Record contains version indicator
					4 FRACHECK preprocessing exit routine, ICHRFX01, is active
					5 RACLIST pre/postprocessing exit routine, ICHRLX01, is active
					6 RACLIST selection exit routine, ICHRLX02, is active
					7 RACDEF postprocessing exit routine, ICHRDX02, is active
124(7C)	SMF81AOP	1	binary	RCVTAUOP	Audit options:
					Bit **Meaning When Set**
					0 User class profile changes are being logged
					1 Group class profile changes are being logged
					2 Data set class profile changes are being logged
					3 Tape volume class profile changes are being logged
					4 DASD volume class profile changes are being logged
					5 Terminal class profile changes are being logged
					6 RACF command violations are being logged
					7 SPECIAL user activity is being logged
125(7D)	SMF81A02	1	binary	-	Reserved
126(7E)	SMF81TMO	1	binary	RCVTEROP	Options indicator:
					Bit **Meaning When Set**
					0 Terminal authorization checking is in effect
					1 UACC for undefined terminals is NONE (if not set, UACC=READ)
					2 REALDSN is in effect
					3 JES-XBMALLRACF is in effect
					4 JES-EARLYVERIFY is in effect
					5 JES-BATCHALLRACF is in effect
					6 FRACHECK postprocessing exit routine, ICHRFX02, is active
					7 Reserved
127(7F)	SMF81PIV	1	binary	RCVTPINV	Maximum password interval (0-254)
128(80)	SMF81REL	2	binary	-	Offset from beginning of the record header to the first relocate section
130(82)	SMF81CNT	2	binary	-	Count of the number of relocate sections
132(84)	SMF81VER	1	binary	RCVTVERS	Version Indicator 6 = Version 1, Release 7
133(85)	SMF81QL	8	char	RCVTQUAL	Single-level data set name
Relocate Section:					
+0(0)	SMF81DTP	1	binary	-	Data type (see Table 2, which is earlier in this chapter under record type 80)
+1(1)	SMF81DLN	1	binary	-	Length of data that follows
+2(2)	SMF81DTA	1-255	mixed	-	Data - (see data types 21 and 32 in Table 2, which is earlier in this chapter under record type 80)

Reformatted RACF SMF Records

For sorting purposes, the RACF report writer reformats SMF records (types 20, 30, 80, and 81) and uses these reformatted records as input to the modules that produce the RACF reports. If you want to use the RACF report writer exit routine (ICHRSMFE) to produce additional reports or to add additional record selection criteria, you should familiarize yourself with the layouts of these reformatted records.

Note: Record type 20 (job initiation) is written by IEFSMFIE at job initiation (including TSO logon). Record type 30 (common address space work record) is written by IEFTB721 at normal or abnormal termination of a batch job or step, a TSO session, or a started task, by IEEMB836 at the expiration of an accounting interval if INTERVAL is specified in SMFPRMxx, by IEFSMFIE at the start of a job or at the start of the first step after a warm start, or by IFAEASI at the expiration of an accounting interval for a system address space. Record types 20 and 30 are documented in *System Management Facilities (SMF).*

There are two record types - reformatted process records and reformatted status records.

In RACF releases prior to Release 7, record types 80 and 81 have had a different mapping in each release. This meant that you had to change the exit routines with each RACF release, and that the routines had to accommodate records with different mappings. With RACF Release 7, RACF restructures these record so that:

- All records have a common format, independent of which release created them.

- Future changes to the records can be made without requiring changes to the exit routines (unless you want to process fields that may be added in future releases).

Notes:

1. *RACF Version 1 Release 7 retains all the field names contained in Release 6.*

2. *The layouts of reformatted process and status records are the same up to the record dependent sections.*

Reformatted Process Records

RACF SMF record types 20, 30, and 80 become reformatted process records. These records are variable in length. Note that a RACF SMF record type 80 generated by a SETROPTS or an RVARY command also causes the creation of a reformatted status record.

The layout of the reformatted process record is:

Offset Dec(Hex)	**Field Name**	**Length**	**Format**	**Description**
0(0)	RCDLEN	2	binary	Total record length
2(2)	-	2	binary	Reserved
4(4)	RCDRELNO	1	binary	Release of RACF
5(5)	RCDREFMT	1	binary	Reformat indicator (if this byte is X'00', the record has been reformatted to the RACF Version 1 Release 6/7 format)
6(6)	RCDSYSID	4	EBCDIC	System identification
10(A)	RCDTYPE	1	EBCDIC	Record type (80 decimal)
11(B)	RCDTIME	4	EBCDIC	Time in form HHMMSSTH
15(F)		1	EBCDIC	Reserved
16(10)	RCDDATE	3	packed	Date in form YYDDDF, where F is the sign
19(13)	RCDFIXLN	2	binary	Offset from the start of the record to the first relocate section
21(15)	RCDCOMLN	2	binary	Offset from the start of the record to the record dependent fields
23(17)	RCDCNT	2	binary	Number of relocate segments
25(19)	RCDEVENT	1	binary	Event code
26(1A)	RCDQUAL	1	binary	Event code qualifier
27(1B)	RCD80FLG	1	binary	Descriptor flags: **Bit** **Meaning When Set** 0 This record is for security violations. 1 This record is for a job/step, not a user/group. 2 This record is truncated. 3 This record is for a warning. 4-7 Reserved.
28(1C)		1	binary	Reserved
29(1D)	RCDUSER	8	EBCDIC	Identifier of the user for which this event is recorded (or jobname if the user is not defined to RACF)
37(25)	RCDGROUP	8	EBCDIC	Group to which the user was connected (or stepname if the user is not defined to RACF)
45(2D)	RCDLOGCL	1	binary	Type of event: 1 - LOGON/JOB 2 - Data set access 3 - RACF command
46(2E)	RCDCLASS	8	EBCDIC	Resource class name (see Note 1). This field contains binary zeroes for records written by the RVARY and SETROPTS commands.

Offset Dec(Hex)	Field Name	Length	Format	Description
54(36)	RCDNAME	44	EBCDIC	Resource name (see Note 1). This field contains the userid for a LOGON/JOB; the resource name for a resource access.
98(62)	RCDJOBID	8	EBCDIC	Job name
106(6A)		1	EBCDIC	Reserved
107(6B	RCDDATID	3	packed	Date that the reader recognized the JOB card for this job in the form YYDDDF
110(6E)	RCDTIMID	4	EBCDIC	Time that the reader recognized the JOB card for this job in the form HHMMSSTH
114(72)	RCDUSRDA	8	EBCDIC	User identification field
122(7A)	RCD80TRM	8	EBCDIC	Terminal identification field
130(82)	RCD80TML	1	binary	Terminal level number
131(83)	RCDOWNER	8	EBCDIC	Owner of the resource

For process records, the record dependent section is:

Offset Dec(Hex)	Field Name	Length	Format	Description
0(0)	RCD80ATH	1	binary	Authority used: **Bit** **Meaning When Set** 0 Normal authority 1 SPECIAL attribute 2 OPERATIONS attribute 3 AUDITOR attribute 4 Exit routine granted authority 5 Failsoft processing 6-7 Reserved
1(1)	RCD80REA	1	binary	Reason for logging: **Bit** **Meaning When Set** 0 Class being audited 1 User being audited 2 Special user being audited 3 Resource being audited, installation-requested logging in effect, or failsoft processing 4 RACINIT failures being audited 5 Command always causes auditing 6 Command violations being audited 7 Audited because GLOBALAUDIT option in effect
2(2)	RCD80ERR	1	binary	Error indicators: **Bit** **Meaning When Set** 0 Command could not recover 1 Profile not altered 2-7 Reserved
3(3)	RCDQUAL1	8	EBCDIC	Qualifier for old data set name (see Note 2)
11(B)	RCDQUAL2	8	EBCDIC	Qualifier for new data set name (see Note 3)
19(13)	RCDDLEV	1	binary	Data set level number (see Note 4)
20(14)	RCDDINT	1	binary	Access authority requested: (see Note 4) **Bit** **Access Authority** 0 ALTER 1 CONTROL 2 UPDATE 3 READ 4 NONE
21(15)	RCDDALWD	1	binary	Access authority allowed: (see Note 4) **Bit** **Access Authority** 0 ALTER 1 CONTROL 2 UPDATE 3 READ 4 NONE
22(16)	RCDDVOL	6	EBCDIC	Volume serial (see Note 4)
28(1C)	RCDDOLDV	6	EBCDIC	OLDVOL volume serial (see Note 4)
34(22)	RCD80GNS	1	binary	1 = Generic name specified

Offset Dec(Hex)	Field Name	Length	Format	Description
35(23)	RCD80GSP	1	binary	1 = Generic name specified on FROM keyword of PERMIT
36(24)	RCD80RRF	1	binary	1 = The **old** name of the RACDEF-renamed data set from data type 33 relocate section
37(25)	RCD80RRT	1	binary	1 = The **new** name of the RACDEF-renamed data set from data type 33 relocate section
38(26)	RCDGENAM	44	EBCDIC	Generic profile used or generic resource name
82(52)	RCDGNNMF	44	EBCDIC	Generic profile used on RACDEF RENAME or generic resource name on RACDEF RENAME
Relocate Section: (See Note 5)				
+0(0)	RCDDTYPE	1	binary	Data type
+1(1)	RCDDLGT	1	binary	Length of data that follows
+2(2)	RCDDATA	variable	mixed	Data

Note 1: In order to support sorting by resource class name and resource name for the list report, the RACF report writer ensures that these fields contain valid names. The following table indicates the resource class names and the resource names assigned by the RACF report writer for each of the event codes in RCDEVENT. (Uppercase letters indicate that the value appears as shown, lowercase letters identify the field in the SMF type 80 record from which the name is obtained, and a number in parentheses identifies the relocate section in the SMF type 80 record from which the name is obtained.)

If RCDEVENT is	Resource Class Name	Resource Name
1	USER	userid (SMF80USR)
2	class name (17)	resource name (1)
3	class name (17)	resource name (1)
4	class name (17)	resource name (1)
5	class name (17)	resource name (1)
6	class name (17)	resource name (1)
7	class name (17)	resource name (1)
8	DATASET	data set name (6)
9	GROUP	group name (6)
10	USER	userid (6)
11	DATASET	data set name (6)
12	GROUP	group name (6)
13	USER	userid (6)
14	USER	userid (6)
15	DATASET	data set name (6)
16	GROUP	group name (6)
17	USER	userid (6)
18	USER	userid (6)
19	class name (17)	resource name (9)
20	class name (17)	resource name (9)
21	class name (17)	resource name (9)
22	class name (17)	resource name (9)
23	USER	userid (6)
24	none	none
25	none	none

Note 2: The RACF report writer compares this field to the DSQUAL keyword specified on the EVENT subcommand. The report writer initializes RCDQUAL1 to the high-level qualifier of the old data set name found in RCDNAME at offset 41 (29 hex) of this record. The RACF report writer exit routine, ICHRSMFE, can modify this field.

Note 3: The RACF report writer compares this field to the NEWDSQUAL keyword specified on the EVENT subcommand. The report writer initializes RCDQUAL to the high-level qualifier of the new data set name found in the relocate section for data type 2 (SMF80DTP = 2). The RACF report writer exit routine, ICHRSMFE, can modify this field.

Note 4: This field is present for event codes 2-7 (SMF80EVT=2 through SMF80EVT=7) only.

Note 5: A relocate section may exist for any of the following event codes (SMF80EVT) and data types (SMF80DTP) combinations:

Event Code	Data Type
1	20
2	20, 33
3	33
4	2, 33
5	33
6	none
7	18, 19
8	6, 10
9	6, 37
10	6, 7, 8, 28, 37
11	6, 10, 11
12	6, 37
13	6, 7, 8, 28, 37
14	6
15	6
16	6
17	6
18	6
19	6, 12, 13, 14, 17, 26, 33
20	6, 7, 10, 11, 17, 24, 25, 29
21	6, 7, 17, 24, 29
22	6, 17
23	6
24	6, 21, 22, 23, 27, 32, 34, 35, 36
25	6, 26

See Table 1 and Table 2 earlier in this chapter for a further explanation of these event codes and data types.

Reformatted Status Records

RACF SMF record types 80 (only those generated by the SETROPTS or RVARY command) and 81 become reformatted status records.

The layout of the reformatted status record is:

Offset **Dec(Hex)**	**Field** **Name**	**Length**	**Format**	**Description**
0(0)	RCDLEN	2	binary	Total record length
2(2)	-	2	binary	Reserved
4(4)	RCDRELNO	1	binary	Release of RACF
5(5)	RCDREFMT	1	binary	Reformat indicator (if this byte is X'00', the record has been reformatted to the RACF Version 1 Release 7 format)
6(6)	RCDSYSID	4	EBCDIC	System identification
10(A)	RCDTYPE	1	EBCDIC	Record type (80 or 81 decimal)
11(B)	RCDTIME	4	EBCDIC	Time in form HHMMSSTH
15(F)		1	EBCDIC	Reserved
16(10)	RCDDATE	3	packed	Date in form YYDDDF, where F is the sign
19(13)	RCDFIXLN	2	binary	Offset from the start of the record to the first relocate section
21(15)	RCDCOMLN	2	binary	Offset from the start of the record to the record dependent fields
23(17)	RCDCNT	2	binary	Number of relocate segments
25(19)	RCDEVENT	1	binary	Event code
26(1A)	RCDQUAL	1	binary	Event code qualifier
27(1B)	RCD80FLG	1	binary	Descriptor flags: **Bit Meaning When Set** 0 This record is for security violations. 1 This record is for a job/step, not a user/group. 2 This record is truncated. 3 This record is for a warning. 4-7 Reserved.
28(1C)		1	binary	Reserved
29(1D)	RCDUSER	8	EBCDIC	Identifier of the user for which this event is recorded (or jobname if the user is not defined to RACF)
37(25)	RCDGROUP	8	EBCDIC	Group to which the user was connected (or stepname if the user is not defined to RACF)
45(2D)	RCDLOGCL	1	binary	Type of event: 1 - LOGON/JOB 2 - Data set access 3 - RACF command

Offset Dec(Hex)	Field Name	Length	Format	Description
46(2E)	RCDCLASS	8	EBCDIC	Resource class name (see Note 1). This field contains binary zeroes for records written by the RVARY and SETROPTS commands.
54(36)	RCDNAME	44	EBCDIC	Resource name (see Note 1). This field contains the userid for a LOGON/JOB; the resource name for a resource access.
98(62)	RCDJOBID	8	EBCDIC	Job name
106(6A)		1	EBCDIC	Reserved
107(6B	RCDDATID	3	packed	Date that the reader recognized the JOB card for this job in the form YYDDDF
110(6E)	RCDTIMID	4	EBCDIC	Time that the reader recognized the JOB card for this job in the form HHMMSSTH
114(72)	RCDUSRDA	8	EBCDIC	User identification field
122(7A)	RCD80TRM	8	EBCDIC	Terminal identification field
130(82)	RCD80TML	1	binary	Terminal level number
131(83)	RCDOWNER	8	EBCDIC	Owner of the resource

For status records, the record dependent section is:

Offset Dec(Hex)	Field Name	Length	Format	Description
0(0)	RCDRACFD	44	EBCDIC	Data set name of the RACF data set for this IPL
44(2C)	RCDRACFV	6	EBCDIC	Volume identification of RACF data set
50(32)	RCDRACFU	3	EBCDIC	Unit name of RACF data set
53(35)	RCD81FLG	1	binary	Options indicators: **Bit** **Meaning When Set** 0 No RACINIT statistics are recorded 1 No DATASET statistics are recorded 2 RACINIT preprocessing exit routine, ICHRIX01, is active 3 RACHECK preprocessing exit routine, ICHRCX01, is active 4 RACDEF preprocessing exit routine, ICHRDX01, is active 5 RACINIT postprocessing exit routine, ICHRIN02, is active 6 RACHECK postprocessing exit routine, ICHRCX02, is active 7 New password exit routine, ICHPWX01, is active
54(36)	RCDUVOL	6	EBCDIC	Volume identification of UADS data set
60(3C)	RCDUDSN	44	EBCDIC	Data set name of the UADS data set for this IPL
104(68)	RCD81FG2	1	binary	Options indicators: **Bit** **Meaning When Set** 0 No tape volume statistics are recorded 1 No DASD volume statistics are recorded 2 No terminal statistics are recorded 3 Command exit routine ICHCNX00 is active 4 Command exit routine ICHCCX00 is active 5 ADSP is not active 6 Encryption exit routine, ICHDEX01, is active 7 Naming convention table, ICHNCV00, is present

Offset Dec(Hex)	Field Name	Length	Format	Description
105(69)	RCD81OP3	1	binary	Options indicators: **Bit** **Meaning When Set** 0 Tape volume protection in effect 1 No duplicate data set protection in effect 2 DASD volume protection in effect 3 Reserved 4 FRACHECK exit routine is active 5 RACLIST pre/postprocessing exit routine is active 6 RACLIST selection exit routine is active 7 RACDEF postprocessing exit routine is active
106(6A)	RCD81AOP	1	binary	Options indicators: **Bit** **Meaning When Set** 0 Log all users 1 Log all groups 2 Log data set class 3 Log tape volume class 4 Log DASD volume class 5 Log terminal class 6 Log command violations 7 Log special users
107(6B)	RCD81TMO	1	binary	Options indicators: **Bit** **Meaning When Set** 0 Terminal authorization checking in effect 1 UACC for undefined terminals is NONE 2 REALDSN is in effect 3 JES-XBMALLRACF is in effect 4 JES-EARLYVERIFY is in effect 5 JES-BATCHALLRACF is in effect 6 FRACHECK postprocessing exit is active 7 Reserved
108(6C)	RCD81PIV	1	binary	Maximum password interval
109(6D)	RCD81MFG	1	binary	Model flags: **Bit** **Meaning When Set** 0 Model - GDG 1 Model - USER 2 Model - GROUP 3-7 Reserved
110(6E)	RCD81MSF	1	binary	Miscellaneous processing flags: **Bit** **Meaning When Set** 0 GRPLIST active 1 Generic profile checking in effect for data set 2 GENCMD in effect for data set class 3 ADSP attribute bypassed 4-7 Reserved

Offset Dec(Hex)	Field Name	Length	Format	Description
111(6F)	RCD81IFG	1	binary	Internal processing flags: **Bit Meaning When Set** 0 The SETROPTS command caused RACFRW to generate this record. 1 The RVARY command caused RACFRW to generate this record. 2 RACF was varied active by RVARY command. 3 This record is incomplete (truncated). 4 RVARY SWITCH was issued. 5-7 Reserved.
112(70)	RCD81CNT	2	binary	Number of relocate sections (see note)
114(72)	RCD81QL	8	char	Single level data set name prefix
Relocate Section:				
+0(0)	RCDDTYPE	1	binary	Data type
+1(1)	RCDDLGT	1	binary	Length of data that follows
+2(2)	RCDDATA	variable	mixed	Data

Note: Only data types (SMF80DTP) 21, 30, 32, 34, 35 or 36 can generate a relocate section for a reformatted status record. See Table 2 earlier in this chapter for a further explanation of these data types.

Chapter 12. RACF Data Areas

ACEE

Common Name: Accessor Environment Element
Macro ID: IHAACEE
DSECT Name: ACEE
Created by: RACINIT SVC 131
Subpool and Key: Subpool 255 and key 0, or subpool specifications by issuer of RACINIT (may reside above 16M)
Size: 144 bytes
Pointed to by: ASXBSENV and TCBSENV, or a field supplied by issuer of RACINIT. (ACEE pointed to by ASXBSENV always resides above 16M)
Serialization: None
Function: This mapping macro maps the RACF Accessor Environment Element or equivalent. This control block represents the authorities of a single accessor in the address space.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	144	ACEE	ACCESSOR ENVIRONMENT ELEMENT
0	(0)	4	ACEEACEE	ACRONYM IN EBCDIC ACEE-
4	(4)	4	ACEECORE	ACEE SUBPOOL AND LENGTH
4	(4)	1	ACEESP	ACEE SUBPOOL NUMBER
5	(5)	3	ACEELEN	LENGTH OF ACEE
8	(8)	1	ACEEVRSN	VERSION = 1.
9	(9)	3		RESERVED
12	(C)	4	ACEEIEP	RESERVED FOR INSTALLATION
16	(10)	4	ACEEINST	ADDRESS OF INSTALLTION SUPPLIED USER DATA FROM USER ENTRY
20	(14)	9	ACEEUSER	USERID INFORMATION
20	(14)	1	ACEEUSRL	USERID LENGTH
21	(15)	8	ACEEUSRI	USERID
29	(1D)	9	ACEEGRP	GROUP NAME INFORMATION
29	(1D)	1	ACEEGRPL	GROUP NAME LENGTH
30	(1E)	8	ACEEGRPN	CONNECT GROUP NAME
38	(26)	1	ACEEFLG1	USER FLAGS
	1...		ACEESPEC	1 SPECIAL ATTRIBUTE
	.1..		ACEEADSP	1 AUTOMATIC DATA SECURITY PROTECTION
	..1.		ACEEOPER	1 OPERATIONS ATTRIBUTE
	...1		ACEEAUDT	1 AUDITOR ATTRIBUTE
	.... 1...		ACEELOGU	1 USER IS TO HAVE MOST RACF FUNCTIONS LOGGED
	.... .1..			RESERVED

OFFSETS		LENGTH	NAME	DESCRIPTION
		.... ..1.	ACEEPRIV	1 USER IS A STARTED PROCEDURE WITH THE PRIVELEGED ATTRIBUTE
		1	ACEERACF	1 RACF DEFINED USER
39	(27)	1	ACEEFLG2	DEFAULT UNIVERSAL ACCESS
		1...	ACEEALTR	1 ALTER AUTHORITY TO RESOURCE
		.1..	ACEECNTL	1 CONTROL AUTHORITY TO RESOURCE
		..1.	ACEEUPDT	1 UPDATE AUTHORITY TO RESOURCE
		...1	ACEEREAD	1 READ AUTHORITY TO RESOURCE
		.... 1...		RESERVED
		.... .1..		RESERVED
		.... ..1.		RESERVED
		1	ACEENONE	1 NO AUTHORITY TO RESOURCE
40	(28)	1	ACEEFLG3	MISCELLANEOUS FLAGS
		1...	ACEEGRPA	ACCESS LIST OF GROUP DS TO CONTAIN 0 USERID 1 GROUP NAME AND USERID
		.1..		RESERVED
		..1.		RESERVED
		...1		RESERVED
		.... 1...		RESERVED
		.... .1..		RESERVED
		.... ..1.		RESERVED
		1		RESERVED
41	(29)	3	ACEEDATE	DATE OF RACINIT
44	(2C)	8	ACEEPROC	NAME OF STARTED PROC OR BLANKS IF NOT STARTED PROC
52	(34)	4	ACEETRMP	ADDRESS OF TERMINAL RELATED INFORMATION. ZERO FOR NON- TERMINAL USERS
56	(38)	2	ACEEFLG4	MISCELLANEOUS FLAGS 2
		1...		RESERVED
		.1..		RESERVED
		..1.	ACEEUATH	1 USER IS AUTHORIZED TO DEFINE OTHER USERS
		...1		RESERVED
		.... 1...	ACEEDASD	1 USER IS AUTHORIZED TO PROTECT DASD VOLUMES
		.... .1..	ACEETAPE	1 USER IS AUTHORIZED TO PROTECT TAPE VOLUMES
		.... ..1.	ACEETERM	1 USER IS AUTHORIZED TO PROTECT TERMINALS
		1		
		1111 1111		RESERVED
58	(3A)	1	ACEEAPLV	APPLICATION LEVEL NUMBER
59	(3B)	1	ACEETRLV	TERMINAL LEVEL NUMBER
60	(3C)	4	ACEETRDA	ADDRESS OF INSTALLATION SUPPLIED DATA FROM TERMINAL ENTRY
64	(40)	8	ACEETRID	TERMINAL ID

OFFSETS		LENGTH	NAME	DESCRIPTION
72	(48)	4	ACEEAMP	ADDRESS FIRST ANCHORED MODEL
76	(4C)	4	ACEECLTH	USER CLASS AUTHORIZATIONS THESE BIT POSITIONS ARE MAPPED BY THE CLASS DESCRIPTOR ENTRIES ANCHORED OFF THE RACF CVT
80	(50)	4	ACEECLCP	ANCHOR FOR INSTORAGE PROFILE TREES BUILT BY THE RACLIST FUNCTION
84	(54)	4	ACEEAPTR	ADDRESS FIELD RESERVED FOR APPLICATION USAGE
88	(58)	8	ACEEAPLN	NAME OF APPLICATION TO WHICH USER IS CONNECTED OR BLANKS IF NO APPLICATION SPECIFIED
96	(60)	4	ACEEAPDA	ADDRESS INSTALLATION SUPPLIED DATA FROM APPLICATION ENTRY
100	(64)	4	ACEEUNAM	ADDRESS OF USER NAME STRING. ZERO, IF NO NAME PRESENT. IF PRESENT, THE FIRST BYTE IS A LENGTH FIELD FOLLOWED BY THE NAME STRING.
104	(68)	4	ACEEMDLS	ADDRESS OF THE DATA SET MODEL NAME ARRAY. ZERO, IF ARRAY NOT OBTAINED BY RACINT.
108	(6C)	4	ACEECGRP	ADDRESS OF TABLE CONTAINING THE LIST OF GROUPS THIS USERID IS A MEMBER OF.
112	(70)	4	ACEEGATA	ADDRESS OF THE GENERIC ANCHOR TABLE
116	(74)	4	ACEEFCGP	ADDRESS OF TABLE CONTAINING THE LIST OF GROUPS THIS USERID IS A MEMBER OF, BUILT BY RACINIT, USED BY FRACHECK, IT IS NOT AUTOMATICALLY REFRESHED
120	(78)	4	ACEEDSLP	ADDRESS OF THE LIST OF DEPARTMENTS TO WHICH THIS USER IS ALLOWED ACCESS
124	(7C)	4	ACEECPRG	ADDRESS OF THE LIST OF CONTROLLED PROGRAMS EXECUTED BY THIS USER
128	(80)	4	ACEEPADS	ADDRESS OF THE LIST OF DATA SETS ACCESSED BY CONTROLLED PROGRAMS EXECUTED BY THIS USER
132	(84)	1	ACEESLVL	MAXIMUM SECURITY LEVEL ACCESSED BY THIS USER
133	(85)	1	ACEEFLG5	MISCELLANEOUS FLAGS
	1...		ACEEMODE	1 ACEE MODE IS IN 31-BIT MODE
	.111 1111			RESERVED

OFFSETS		LENGTH	NAME	DESCRIPTION
134	(86)	2		RESERVED.
136	(88)	8		RESERVED.

BAMTAB

Common Name: BAM Locator Block
Macro ID: ICHPBAMP
DSECT Name: BAMTAB
Created by: ICHSEC00
Subpool and Key: 241 and key 0
Size: variable
Pointed to by: RCVTBAM field of the .... data area
Serialization: None
Function: This control block contains locators (RBA) of every BAM block in the RACF data set. It allows for locating the necessary BAM block used in space allocation (for extending an entry), or space deallocation (for entry deletion), without any I/O processing to the RACF data set.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	8	BAMTAB	LOCATED FROM RACF CVT
0	(0)	4		RESERVED
4	(4)	4	BAMNUM	NUMBER OF BAM BLOCKS IN RACF DATA SET
8	(8)	0	BAMRBAS	RBA LOCATORS OF EACH BAM BLOCK

CGRP

Common Name: Connect Group Name Table Definition

Macro ID: ICHPCGRP

DSECT Name: CGRP, CGRPENTD

Created by: RACINIT, RACHECK

Subpool and Key: 255 and key 0

Size: Variable, Fixed 32 bytes + 24 bytes per connect group

Pointed to by: ACEECGRP or ACEEFCGP field of the ACEE data area

Serialization: local lock

Function: This table contains the names of the groups that the ACEEUSRI userid is a member of.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	32	CGRP	CONNECT GROUP NAME TABLE.
0	(0)	4	CGRPID	TABLE ID.
4	(4)	4	CGRPCORE	CGRP SUBPOOL AND LENGTH.
4	(4)	1	CGRPSP	SUBPOOL NUMBER.
5	(5)	3	CGRPLEN	LENGTH OF CGRP.
8	(8)	2	CGRPNUM	MAXIMUM ENTRIES IN TABLE.
10	(A)	1	CGRPVRSN	VERSION = 1.
11	(B)	1		RESERVED.
12	(C)	4	CGRPSYNC	SYNCRONIZE VALUE.
16	(10)	4	CGRPGPAT	ADDRESS OF GROUP AUTHORITIES TABLE, OR ZERO IF NO SUCH TABLE EXISTS
20	(14)	12		RESERVED
32	(20)	0	CGRPENT	GROUP NAME ENTRY.
32	(20)	8	CGRPNAME	GROUP NAME.
40	(28)	1	CGRPIND	INDICATORS FOR THIS ENTRY
	1...		CGRPCHK	ALWAYS ZERO, WAS REVOKE INDICATOR
	.1..		CGRPREFR	ON IF GROUP AUTHORITY TABLE MUST BE REFRESHED FOR THIS CONNECT GROUP
	..1.		CGRPCOMP	ON IF GROUP ENTERED INTO GROUP AUTHORITY TABLE AND NO LATER AUTHORITY CHANGES WERE MADE OR THE GROUP DID NOT NEED TO BE ENTERED INTO THE TABLE
	...1		CGRPPROP	ON IF THIS GROUP IS OWNED BY ITS SUPERIOR GROUP. INDICATES THE GROUP IS PART OF THE SUBGROUP TREE FOR PROPAGATION OF GROUP AUTHORITIES
	 1111			RESERVED

OFFSETS		LENGTH	NAME	DESCRIPTION
41	(29)	1	CGRPAUTH	GROUP AUTHORITY INDICATORS
		1...	CGRPSPEC	ON IF GROUP-SPECIAL AUTHORITY
		.1..		RESERVED
		..1.	CGRPOPER	ON IF GROUP-OPERATIONS AUTHORITY
		...1	CGRPAUDT	ON IF GROUP-AUDITOR AUTHORITY
		 1111		RESERVED
42	(2A)	2	CGRPGPNM	NUMBER OF ENTRIES IN GROUP AUTHORITY TABLE RELATED TO THIS CONNECT GROUP
44	(2C)	4	CGRPGPTE	ADDRESS OF FIRST GROUP AUTHORITY TABLE ENTRY RELATED TO THIS CONNECT GROUP
48	(30)	2	CGRPSUPG	INDEX IN CGRPENT OF ENTRY FOR SUPERIOR GROUP OF THIS ENTRY, TO WHICH THE USER IS CONNECTED
50	(32)	6		RESERVED.
56	(38)	0		END OF ENTRY.

CNST

Common Name: RACF Class Descriptor Table Entry
Macro ID: ICHPCNST
DSECT Name: CNST
Created by: RACF Initialization - MSI
Subpool and Key: 252 and key 0
Size: 28 bytes/class entry
Pointed to by: RCVTCDTP field of the RCVT data area.
Serialization: None
Function: This table describes a general resource class to RACF. It contains the name of the general class, the resource name syntax and control information. There is one entry for each general resource class.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	28	CNST	CLASS NAME/SYNTAX TABLE
0	(0)	2	CNSTLGT	LENGTH OF ENTRY
2	(2)	1	CNSTID	CLASS ID NUMBER
3	(3)	8	CNSTNAME	CLASS NAME
11	(B)	8	CNSTXREF	GROUP/MEMBER CLASS NAME
19	(13)	3	CNSTSNTX	CLASS MEMBER NAME SYNTAX
19	(13)	1	CNSTMAXL	MAX LENGTH OF MEMBER NAME
20	(14)	1	CNSTFRST	SYNTAX OF FIRST CHARACTER
		1...	CNSTFALP	1 => ALPHABETIC CHAR ALLOWED
		.1..	CNSTFNAT	1 => NATIONAL CHAR ALLOWED
		..1.	CNSTFNUM	1 => NUMERIC CHAR ALLOWED
		...1	CNSTFSPE	1 => SPECIAL CHAR ALLOWED
		 1111		RESERVED
21	(15)	1	CNSTREMN	SYNTAX OF REMAINING CHARACTER
		1...	CNSTRALP	1 => ALPHABETIC CHAR ALLOWED
		.1..	CNSTRNAT	1 => NATIONAL CHAR ALLOWED
		..1.	CNSTRNUM	1 => NUMERIC CHAR ALLOWED
		...1	CNSTRSPE	1 => SPECIAL CHAR ALLOWED
		 1111		RESERVED
22	(16)	1	CNSTUACC	DEFAULT UACC
		1...	CNSTALTR	1 => ALTER UACC
		.1..	CNSTCNTL	1 => CONTROL UACC
		..1.	CNSTUPDT	1 => UPDATE UACC
		...1	CNSTREAD	1 => READ UACC
		.... 111.		RESERVED
		1	CNSTNONE	1 => NONE UACC
23	(17)	1	CNSTMFLG	MISC. FLAGS
		1...	CNSTRGRP	1 => CLASS IS RESOURCE GROUP
		.1..	CNSTACEE	1 => USE UACC FROM ACEE
		..1.	CNSTOPER	1 => OPERATIONS ATTRIB. APPLIES TO THIS CLASS

OFFSETS		LENGTH	NAME	DESCRIPTION
		...1 1111		RESERVED
24	(18)	4	CNSTMASK	MASK FOR OPTION FLAGS

DSDT

Common Name: RACF Data Set Descriptor Table
Macro ID: ICHDSDT
DSECT Name: DSDT
Created by: ICHSEC00
Subpool and Key: 241 (CSA or ECSA) and key 0
Size: 8 byte header plus 160 bytes per primary or backup data set pair
Pointed to by: RCVTDSDT field of the RCVT data area.
Serialization: None
Function: This table describes primary and backup RACF data sets.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	8	DSDT	LOCATED THROUGH RCVT
0	(0)	4	DSDTID	EBCDIC ID
4	(4)	4	DSDTNUM	NUMBER OF ENTRIES IN TABLE
8	(8)	0	DSDTENTY	ENTRY FOR DATA SET INFORMATION
8	(8)	80	DSDTPRIM	ENTRY FOR PRIMARY DATA SET
8	(8)	4	DSDPDCB	PTR DCB PRIMARY RACF DATA SET
12	(C)	4	DSDPDEB	PTR DEB PRIMARY RACF DATA SET
16	(10)	4	DSDPINDX	PTR TO IN-STORAGE BUFFERS OR RESIDENT INDEX TABLE. ZERO IF NO RESIDENT BLOCKS FOR THE PRIMARY DATA SET.
20	(14)	4	DSDPHDR	PTR RACF IN-STORAGE DS HEADER RECORD OR ZERO IF PRIMARY RACF DATA SET IS ON A SHARED DEVICE
24	(18)	4	DSDPRUCB	PTR UCB PRIMARY RACF DATA SET
28	(1C)	4	DSDPXLEN	LENGTH OF IN-STORAGE BUFFERS OR RESIDENT INDEX TABLE FOR THE PRIMARY RACF DATA SET
32	(20)	4	DSDPBAM	LOCATES IN-STORAGE BAM INFORMATION FOR PRIMARY DATA SET
36	(24)	1	DSDPDSNL	LENGTH OF PRIMARY RACF DATA SET NAME
37	(25)	1	DSDPSTAT	PRIMARY RACF DATA SET STATUS
		1...	DSDPACTV	THIS DATA SET IS ACTIVE
		.1..	DSDPPRIM	THIS DATA SET IS A PRIMARY
		..1.	DSDPMSTR	THIS DATA SET IS THE MASTER RACF DATA SET. ITS ICB CONTAINS STATUS OPTIONS.
		...1	DSDPRFSH	REFRESH ICB
		.... 1...	DSDPSHR	DATA SET IS (OR WAS) SHARED

OFFSETS		LENGTH	NAME	DESCRIPTION
		.... .1..	DSDPALTI	ALTERI REQUESTS ARE BACKED-UP
		.... ..1.	DSDPDAT	IN-STORAGE BLOCKS CAN BE DATA BLOCKS
38	(26)	2	DSDPNREC	# RECORDS PER TRACK PRIMARY DATA SET
40	(28)	1	DSDPRXNO	# IN-STORAGE BUFFERS OR RESIDENT INDEX BLOCKS
41	(29)	44	DSDPDSN	DSN OF RACF PRIMARY DATA SET
85	(55)	1	DSDPDSNO	DATA SET SEQUENCE NUMBER
86	(56)	2		RESERVED
88	(58)	80	DSDTBACK	ENTRY FOR BACKUP DATA SET
88	(58)	4	DSDBDCB	PTR DCB OF BACK-UP DATA SET
92	(5C)	4	DSDBDEB	PTR DEB OF BACK-UP DATA SET
96	(60)	4	DSDBINDX	PTR TO IN-STORAGE BUFFERS OR RESIDENT INDEX TABLE. ZERO IF NO RESIDENT BLOCKS FOR THE BACK-UP RACF DATA SET
100	(64)	4	DSDBHDR	PTR RACF IN-STORAGE DS HEADER RECORD OR ZERO IF BACK-UP RACF DATA SET IS ON A SHARED DEVICE
104	(68)	4	DSDBRUCB	PTR UCB OF BACK-UP DATA SET
108	(6C)	4	DSDBXLEN	LENGTH OF IN-STORAGE BUFFERS OR RESIDENT INDEX TABLE FOR THE BACK-UP RACF DATA SET
112	(70)	4	DSDBBAM	LOCATES IN-STORAGE BAM INFORMATION FOR BACK-UP DATA SET
116	(74)	1	DSDBDSNL	LENGTH OF BACK-UP DATA SET NAME
117	(75)	1	DSDBSTAT	STATUS OF BACK-UP DATA SET
		1...	DSDBACTV	THIS DATA SET IS ACTIVE
		.1..	DSDBPRIM	THIS DATA SET IS A PRIMARY
		..1.	DSDBMSTR	THIS DATA SET IS THE MASTER RACF DATA SET. ITS ICB CONTAINS STATUS OPTIONS.
		...1	DSDBRFSH	REFRESH ICB
		.... 1...	DSDBSHR	DATA SET IS (OR WAS) SHARED
		.... .1..	DSDBALTI	ALTERI REQUESTS ARE BACKED-UP
		.... ..1.	DSDBDAT	IN-STORAGE BLOCKS CAN BE DATA BLOCKS
118	(76)	2	DSDBNREC	# RECORDS PER TRACK BACK-UP DATA SET
120	(78)	1	DSDBRXNO	# IN STORAGE BUFFERS OR RESIDENT INDEX BLOCKS
121	(79)	44	DSDBDSN	DSN OF BACK-UP RACF DATA SET
165	(A5)	1	DSDBDSNO	DATA SET SEQUENCE NUMBER
166	(A6)	2		RESERVED
0	(0)	80	DSDE	ENTRY FOR DATA SET

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	4	DSDEDCB	PTR DCB FOR DATA SET
4	(4)	4	DSDEDEB	PTR DEB FOR DATA SET
8	(8)	4	DSDEINDX	PTR TO IN-STORAGE BUFFERS OR RESIDENT INDEX TABLE. ZERO IF NO RESIDENT BLOCKS FOR THE DATA SET
12	(C)	4	DSDEHDR	PTR RACF IN-STORAGE DS HEADER RECORD OR ZERO IF DATA SET IS ON A SHARED DEVICE
16	(10)	4	DSDERUCB	PTR UCB FOR DATA SET
20	(14)	4	DSDEXLEN	LENGTH OF IN-STORAGE INDEX RELATED CONTROL BLOCKS FOR DATA SET
24	(18)	4	DSDEBAM	LOCATES IN-STORAGE BAM INFORMATION FOR DATA SET
28	(1C)	1	DSDEDSNL	LENGTH OF DATA SET NAME
29	(1D)	1	DSDESTAT	DATA SET STATUS
		1...	DSDEACTV	THIS DATA SET IS ACTIVE
		.1..	DSDEPRIM	THIS DATA SET IS A PRIMARY
		..1.	DSDEMSTR	THIS DATA SET IS THE MASTER RACF RACF DATA SET. ITS ICB CONTAINS STATUS OPTIONS.
		...1	DSDERFSH	REFRESH ICB
		.... 1...	DSDESHR	DATA SET IS (OR WAS) SHARED
		.... .1..	DSDEALTI	ALTERI REQUESTS ARE BACKED-UP
		.... ..1.	DSDEDAT	IN-STORAGE BLOCKS CAN BE DATA BLOCKS
30	(1E)	2	DSDENREC	# RECORDS/TRACK ON DATA SET
32	(20)	1	DSDERXNO	# IN-STORAGE BUFFERS OR RESIDENT INDEX BLOCKS
33	(21)	44	DSDEDSN	NAME OF DATA SET
77	(4D)	1	DSDEDSNO	DATA SET SEQUENCE NUMBER
78	(4E)	2		RESERVED

FPBAS

Common Name: Global Access Checking Base Table

Macro ID: ICHFPBAS

DSECT Name: FPBAS, FPBENT

Created by: ICHSEC04

Subpool and Key: 245 and key 0

Size: Base segment 12 bytes + 16 bytes for each class

Pointed to by: RCVTFPB field of the RACF CVT data area

Serialization: None

Function: Contains pointers to 'current' and 'old' global access checking tables for each class.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	16	FPBAS	FASTPATH BASE TABLE.
0	(0)	16	FPB	BASE PORTION OF TABLE.
0	(0)	4	FPBID	TABLE IDENTIFIER = FPBT
4	(4)	4	FPBCORE	FPB SUBPOOL AND LENGTH.
4	(4)	1	FPBSP	SUBPOOL NUMBER.
5	(5)	3	FPBLEN	LENGTH OF ICHFPBAS.
8	(8)	2	FPBNUM	NUMBER ENTRIES IN TABLE.
10	(A)	1		RESERVED.
11	(B)	1	FPBVRSN	VERSION = 0.
12	(C)	4		RESERVED.
16	(10)	0	FPBESTRT	START OF ENTRY NAMES.

FPCLS

Common Name: Global Access Checking Class Table
Macro ID: ICHFPCLS
DSECT Name: FPCLS, FPCENTRY
Created by: ICHFPD01
Subpool and Key: 245 and key 0
Size: Variable
Pointed to by: FPBCURPT, FBPOLDPT in FPBAS entries
Serialization: None
Function: Contains profile names for use by global access checking.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	18	FPCLS	FASTPATH CLASS PROFILE TABLE
0	(0)	18	FPC	BASE PORTION OF TABLE.
0	(0)	4	FPCID	TABLE IDENTIFIER = FPCL
4	(4)	4	FPCCORE	SUBPOOL AND SIZE OF TABLE.
4	(4)	1	FPCSP	SUBPOOL NUMBER FOR TABLE.
5	(5)	3	FPCLEN	SIZE OF THIS TABLE.
8	(8)	3		RESERVED.
11	(B)	1	FPCVRSN	VERSION = 0.
12	(C)	4	FPCNUM	NUMBER OF ENTRIES IN TABLE.
16	(10)	2	FPCELEN	TOT LEN OF ENTRY.
18	(12)	0	FPCENT	PROFILE NAME ENTRY.

GAPL

Common Name: Generic Anchor Table Entry
Macro ID: ICHGAPL
DSECT Name: GENATE, GENPRFL, GENPLEL
Created by: ICHGLS00 (LSQA or ELSQA)
Subpool and Key: 255 and key 0
Size: Variable
Pointed to by: RCVTGATA in RACF CVT data area
ATENEXT field in GAPL data area
Serialization: Local Lock
Function: Contains descriptor and generic profile names for general resource class or data set high-level qualifier.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	40	GENATE	ANCHOR TABLE ENTRY (ATE)
0	(0)	4	ATEID	BLOCK IDENTIFYER
4	(4)	1	ATEVERSN	VERSION NUMBER
5	(5)	3		RESERVED
8	(8)	4	ATESPLN	SUBPOOL NUMBER AND LENGTH OF ATE
8	(8)	1	ATESP	SUBPOOL NUMBER
9	(9)	3	ATELN	LENGTH OF ATE
12	(C)	8	ATENM	CLASS NAME OR HLQ IF DATASET
20	(14)	1	ATEFLAG	FLAG BYTE
	1...		ATETYP	CLASS TYPE ON DATASET , OFF GENERAL
	.1..			RESERVED
	..1.			RESERVED
	...1			RESERVED
	.... 1...			RESERVED
	.... .1..			RESERVED
	.... ..1.			RESERVED
	1			RESERVED
21	(15)	3		RESERVED
24	(18)	4		RESERVED
28	(1C)	4	ATERCNT	REFRESH COUNT
32	(20)	4	ATEPROFL	ADDRESS OF PROFILE LIST
36	(24)	4	ATENEXT	ADDRESS OF NEXT ENTRY IN ANCHOR TABLE

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	28	GENPRFL	PROFILE LIST BLOCK HEADER
0	(0)	4	PRFLID	PROFILE LIST BLOCK ID 'GPRF'
4	(4)	4	PRFLSPLN	SUBPOOL NUMBER AND LENGTH OF BLOCK
4	(4)	1	PRFLSP	SUBPOOL NUMBER
5	(5)	3	PRFLLN	LENGTH OF BLOCK
8	(8)	4	PRFLNEXT	ADDRESS OF NEXT BLOCK IN PROFILE LIST
12	(C)	2	PRFLNE	NUMBER OF ENTRIES IN THIS BLOCK
14	(E)	2		RESERVED
16	(10)	2	PRFLLH	LENGTH OF HEADER
18	(12)	2	PRFLLE	IF RACLIST FORMAT (FIXED-LENGTH ENTRIES), LENGTH OF EACH ENTRY IN THE PROFILE LIST. IF NORMAL FORMAT (VARIABLE-LENGTH ENTRIES), ZERO(USE PLELNL AND LENGTH(GENPLEL) INSTEAD
20	(14)	8		RESERVED
28	(1C)	0	PRFENT	START OF PROFILE LIST ELEMENTS IN BLOCK
0	(0)	8	GENPLEL	PROFILE LIST ELEMENT
0	(0)	2	PLELNL	LENGTH OF RESOURCE NAME
2	(2)	1	PLELFLGS	FLAG BITS
	1...		PLELRTRV	ON IF PROFILE RETRIEVAL HAS BEEN DONE (OR ATTEMPTED, SINCE THE MIGHT BE MISSING WHEN THE ATTEMPT IS DONE).
3	(3)	1		RESERVED
4	(4)	4	PLELPRF	ADDRESS OF PROFILE, OR ZERO IF THE PROFILE HAS NOT YET BEEN RETRIEVED (OR IF IT CANNOT BE FOUND). THIS FIELD IS ONLY VALID IF PLELRTRV IS ON.
8	(8)	0	PLELNM	RESOURE NAME

GRPF

Common Name: In-storage Generic Profile Map
Macro ID: ICHGRPF
DSECT Name: GRPF
Created by: ICHGLS00
Subpool and Key: 255 (LSWA or ELSQA) and key 0
Size: Variable
Pointed to by: PLELPRF field in GAPL data area
Serialization: Local Lock
Function: Describes the structure of an in-storage generic profile.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	56	GRPF	GENERIC PROFILE MAP
0	(0)	4	GRPFSPLN	
0	(0)	1	GRPFS	AREA SUBPOOL NUMBER
1	(1)	3	GRPFL	TOTAL AREA LENGTH
4	(4)	52	GRPFST	PROFILE DATA
4	(4)	8	GRPFOWNR	RESOURCE OWNER
12	(C)	1	GRPFUACC	UNIVERSAL ACCESS
13	(D)	1	GRPFAUDT	AUDIT FLAGS
14	(E)	1	GRPFGAUD	GLOBAL AUDIT FLAGS
15	(F)	1	GRPFLEVL	RESOURCE LEVEL
16	(10)	4	GRPFACOF	OFFSET TO ACCESS
20	(14)	4	GRPFINOF	OFFSET TO INSTALLATION DATA
24	(18)	1	GRPFGPIN	GROUP/USER DATASET INDICATOR
25	(19)	1	GRPFWARN	WARNING VALUE
26	(1A)	2	GRPFRTPD	RETENTION PERIOD
28	(1C)	1	GRPFEOS	ERASE FLAG
29	(1D)	1	GRPFSLVL	RESOURCE SECLEVEL
30	(1E)	1	GRPFLDAY	DAYS OF THE WEEK TERMINAL MAY NOT BE USED
31	(1F)	8	GRPFNTFY	USERID TO NOTIFY WHEN THIS PROFILE DENIES ACCESS
39	(27)	3	GRPFLGNT	EARLIEST TIME A TERMINAL MAY BE USED
42	(2A)	3	GRPFLGFT	LATEST TIME A TERMINAL MAY BE USED
45	(2D)	3	GRPFTZNE	TIME ZONE OFFSET OF TERMINAL FROM CPU
48	(30)	4	GRPFDPOF	OFFSET TO RESOURCE CATEGORY LIST
52	(34)	4	GRPFA2OF	OFFSET TO CONDITIONAL ACCESS LIST

```
...... RACF VERSION 1 REL 7     FRAME                    DATA AREA
DATA AREAS                       A31                      GRPF
LICENSED MATERIAL - PROPERTY OF IBM
```

ISP

Common Name: RACF In-Storage Profile
Macro ID: ICHPISP
DSECT Name: RACRTE,RACRSE,RACRNE,RPEINSD,RACRPE,RPEACCLE
Created by: RACLIST SVC
Subpool and Key: 255 or subpool specified by issuer of RACLIST
and key 0 (may reside above 16M)
Size: Variable
Pointed to by: ACEECLCP field of the ACEE data area.
Serialization: None
Function: This table contains profiles for general resources in a class plus control information for locating individual profiles.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	24	RACRTE	RACLIST CLASS TREE ANCHOR ELEMENT
0	(0)	4	RTENEXT	ADDRESS OF NEXT ANCHOR OR 0
4	(4)	4	RTECLASS	ADDRESS OF CLASS DESCRIPTOR ENTRY FOR THIS CLASS
8	(8)	4	RTETREE	ADDRESS OF TOP NODE IN TREE OR 0
12	(C)	4	RTESTORE	ADDRESS OF STORAGE BLOCK LIST OR 0
16	(10)	2	RTESPNS	PROFILE & NODE SUBPOOL NUMBERS
16	(10)	1	RTEPSPN	SUBPOOL NUMBER FOR PROFILES
17	(11)	1	RTENSPN	SUBPOOL NUMBER FOR TREE NODES
18	(12)	1	RTEASPN	SUBPOOL NUMBER OF THIS BLOCK
19	(13)	1		RESERVED
20	(14)	4	RTEGENL	ADDRESS OF GENERIC PROFILE LIST OR 0
0	(0)	8	RACRSE	RACLIST CLASS TREE STORAGE BLOCK
0	(0)	4	RSENEXT	ADDRESS OF NEXT STORAGE BLOCK OR 0
4	(4)	2	RSESIZE	LENGTH OF STORAGE BLOCK
6	(6)	1	RSEPOOL	SUBPOOL NUMBER OF STORAGE BLOCK
7	(7)	1		RESERVED
8	(8)	0	RSESTORE	USEABLE STORAGE (RSESIZE-4 BYTES)
0	(0)	16	RACRNE	RACLIST CLASS TREE NODE ELEMENT
0	(0)	4	RNELEFT	ADDRESS OF LEFT DAUGHTER NODE OR 0
4	(4)	4	RNEPROF	ADDRESS OF PROFILE FOR THIS NODE

OFFSETS		LENGTH	NAME	DESCRIPTION
8	(8)	4	RNERIGHT	ADDRESS OF RIGHT DAUGHTER NODE OR 0
12	(C)	4	RNEBAL	TREE BALANCING FACTOR DURING TREE CREATION
12	(C)	4	RNEUP	POINTER TO MOTHER NODE DURING TREE DELETION
16	(10)	0	RNEKEY	KEY (LEGNTH DETERMINED BY MAXIMUM NAME LENGTH FOR CLASS IN THE CLASS DESCRIPTOR ELEMENT)
0	(0)	52	RACRPE	RESOURCE PROFILE ELEMENT
0	(0)	2	RPEPLEN	PHYSICAL STORAGE LENGTH OF BLOCK
2	(2)	2	RPELLEN	LOGICAL LENGTH OF BLOCK
4	(4)	2	RPEUCNT	NUMBER OF RESOURCES SHARING THIS PROFILE
6	(6)	4	RPEATTR	ATTRIBUTE FLAGS
6	(6)	1	RPEUACC	UNIVERSAL ACCESS
7	(7)	1	RPEAUDIT	AUDIT FLAGS
8	(8)	1	RPEGAUD	GLOBAL AUDIT FLAGS
9	(9)	1	RPELEVEL	RESOURCE LEVEL
10	(A)	2	RPEACCNO	NUMBER OF ENTRIES IN ACCESS LIST
12	(C)	2	RPEACCOF	OFFSET TO ACCESS LIST
14	(E)	2	RPEINSOF	OFFSET TO INSTALLATION DATA
16	(10)	2	RPEAPPOF	OFFSET TO APPLICATION DATA
18	(12)	8	RPEOWNER	OWNER OF RESOURCE PROFILE
26	(1A)	2	RPENUMDP	NUMBER OF CATEGORIES IN LIST
28	(1C)	2	RPEDPTOF	OFFSET TO CATEGORY LIST
30	(1E)	1	RPELDAYS	DAYS TERMINAL MAY NOT BE USED (BIT 0 SUNDAY, BIT 1 MONDAY,)
31	(1F)	1	RPESCLVL	RESOURCE SECURITY LEVEL
32	(20)	3	RPELOGNT	EARLIEST TIME TERMINAL MAY BE USED (HHMM)
35	(23)	3	RPELOGFT	LATEST TIME TERMINAL MAY BE USED (HHMM)
38	(26)	8	RPENTFY	USERID TO NOTIFY WHEN THIS PROFILE DENIES ACCESS
46	(2E)	3	RPETZONE	TIME OFFSET OF TERMINAL FROM CPU. + = EAST = WEST.
49	(31)	3		RESERVED
52	(34)	0	RPEEND	END OF FIXED PART OF ELEMENT
0	(0)	1	RPEINST	INSTALLATION DATA VARIABLE LENGTH PORTION
0	(0)	1	RPEINSTL	INSTALLATION DATA LENGTH
1	(1)	0	RPEINSTD	INSTALLATION DATA STRING
0	(0)	1	RPEAPPL	APPLICATION DATA VARIABLE LENGTH PORTION
0	(0)	1	RPEAPPLL	APPLICATION DATA LENGTH

OFFSETS		LENGTH	NAME	DESCRIPTION
1	(1)	0	RPEAPPLD	APPLICATION DATA STRING
0	(0)	0	RPEACCLE	ACCESS LIST
0	(0)	8	RPEAUSR	USER/GROUP ID
8	(8)	1	RPEACS	ACCESS AUTHORITY

MDEL

Common Name: Data Set Model Name Table Definition
Macro ID: ICHPMDEL
DSECT Name: MDEL
Created by: RACINIT, RACDEF
Subpool and Key: 255 or subpool supplied by issuer of racinit and key 0
Size: 65 bytes
Pointed to by: ACEEMDLS field of the ACEE data area
Serialization: None
Function: This table contains the names of the models that have been accessed during a job or session. able.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	0	MDEL	MODEL NAME TABLE
0	(0)	4	MDELID	TABLE IDENTIFIER = MDEL
4	(4)	4	MDELCORE	MDEL SUBPOOL AND LENGTH.
4	(4)	1	MDELSP	SUBPOOL NUMBER FOR TABLE.
5	(5)	3	MDELLEN	LENGTH OF MDEL TABLE.
8	(8)	2	MDELNUM	MAXIMUM NUMBER OF ENTRIES IN THIS TABLE.
10	(A)	1		RESERVED.
11	(B)	1	MDELVRSN	VERSION = 0.
12	(C)	4	MDELNXT	ADDRESS OF NEXT MDEL TABLE OR ZERO IF NONE.
16	(10)	49	MDELENT	MODEL NAME ENTRY.

FOLLOWING DSECT DESCRIBES AN INDIVIDUAL MODEL NAME ENTRY.

0	(0)	0	MDELENTD	DESCRIPTION OF INDIVIDUAL ENTRY.
0	(0)	1	MDELIND	INDICATORS FOR THIS ENTRY.
	1...	....	MDELABS	"X'80'" MODEL PROFILE IS ABSENT.
	.1..	....	MDELUSR	"X'40'" MODEL ENTRY FOR THE USER.
	..1.	....	MDELCHK	"X'20'" MODEL HAS BEEN LOOKED FOR.
1	(1)	1	MDELLOCK	IN USE INDICATOR. WILL BE ZERO IF FREE AND X'FF' IF IN USE.
2	(2)	2		RESERVED.
4	(4)	1	MDELQULL	NONBLANK LENGTH OF QUALIFIER.
5	(5)	8	MDELQUAL	FIRST LEVEL QUALIFIER.
13	(D)	1	MDELREML	NONBLANK LENGTH OF REMAINDER OF MODEL NAME.

OFFSETS		LENGTH	NAME	DESCRIPTION
14	(E)	35	MDELREM	RELATIVE MODEL NAME.

RCVT

Common Name: RACF Communication Vector Table
Macro ID: ICHPRCVT
DSECT Name: RCVT
Created by: RACF ICHSEC00 or equivalent
Subpool and Key: SQA and key 0
Size: 428 bytes
Pointed to by: CVTRAC
Serialization: None
Function: Communication area for information global to RACF functions or equivalent

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	512	RCVT	LOCATED THROUGH CVT
0	(0)	4	RCVTID	EBCDIC ID
4	(4)	4	RCVTDCB	PTR DCB OF RACF DATA SET
8	(8)	4	RCVTDEB	PTR DEB OF RACF DATA SET
12	(C)	4	RCVTINDX	PTR RACF RESIDENT INDEX TABLE OR ZERO IF NO INDEX BLOCKS RESIDENT
16	(10)	4	RCVTTEMP	PTR RACF INCORE TEMPLATE TABLE
20	(14)	4	RCVTHDR	PTR RACF INCORE DS HEADER RECORD OR ZERO IF RACF DATA SET IS ON A SHARED DEVICE
24	(18)	4	RCVTRIX	PTR RACINIT INSTALL. EXIT RTN
28	(1C)	4	RCVTRCX	PTR RACCHK INSTALL. EXIT RTN
32	(20)	4	RCVTRDX	PTR RACDEF INSTALL. EXIT RTN
36	(24)	4	RCVTRUCB	PTR UCB OF RACF DATA SET
40	(28)	4	RCVTXLEN	LENGTH OF INCORE INDEX RELATED CONTROL BLOCKS
44	(2C)	4	RCVTBAM	LOCATES INCORE BAM INFORMATION
48	(30)	4	RCVTISTL	RESERVED FOR INSTALLATION
52	(34)	1	RCVTDSNL	LENGTH OF RAC DATA SET NAME
53	(35)	1	RCVTSTAT	STATUS
		1...	RCVTRNA	RACF NOT ACTIVE

```
 OFFSETS  LENGTH  NAME        DESCRIPTION
         .1.. ....  RCVTNLS     BYPASS RACINIT STATISTICS
         ..1. ....  RCVTNDSS    BYPASS DATA SET STATISTICS
         ...1 ....  RCVTNTVS    NO TAPE VOLUME STATISTICS
         .... 1...  RCVTNDVS    NO DIRECT ACCESS VOLUME STATISTICS
         .... .1..  RCVTNTMS    NO TERMINAL STATISTICS
         .... ..1.  RCVTNADS    NO ADSP PROTECTION
         .... ...1              RESERVED
  54    (36)     2  RCVTNREC    # RECORDS PER TRACK RACF DS
------------------------------------------------------------------------
  56    (38)    44  RCVTDSN     DSN OF RACF DATA SET
------------------------------------------------------------------------
 100    (64)    44  RCVTUADS    DSN OF UADS DATA SET OR ZERO
------------------------------------------------------------------------
 144    (90)     6  RCVTUVOL    VOLID OF UADS DATA SET OR ZERO
 150    (96)     1  RCVTSTA1
         1... ....  RCVTTAPE    TAPE VOLUME PROTECTION IN EFFECT
         .1.. ....  RCVTDASD    DASD VOLUME PROTECTION IN EFFECT
         ..1. ....  RCVTDGEN    GENERIC PROFILE CHECKING IN EFFECT FOR DATASET CLASS
         ...1 ....  RCVTDGCM    GENERIC COMMAND PROCESSING IN EFFECT FOR DATASET
                                CLASS
         .... 1...  RCVTRDSN    INPUT DATA SET NAME WILL BE USED FOR LOGGING AND
                                MESSAGES
         .... .1..  RCVTJXAL    JES-XBMALLRACF IN EFFECT
         .... ..1.  RCVTJCHK    JES-EARLYVERIFY IN EFFECT
         .... ...1  RCVTJALL    JES-BATCHALLRACF IN EFFECT
 151    (97)     1  RCVTAUOP    RACF AUDIT OPTIONS
         1... ....              RESERVED
         .1.. ....  RCVTAGRO    AUDIT GROUP CLASS
         ..1. ....  RCVTAUSE    AUDIT USER CLASS
         ...1 ....  RCVTADAT    AUDIT DATASET CLASS
         .... 1...  RCVTADAS    AUDIT DASDVOL CLASS
         .... .1..  RCVTATAP    AUDIT TAPEVOL CLASS
         .... ..1.  RCVTATER    AUDIT TERMINAL CLASS
         .... ...1  RCVTAOPR    AUDIT OPERATIONS ATTRIBUTE
------------------------------------------------------------------------
 152    (98)     1  RCVTAXTA    RESERVED
 153    (99)     1  RCVTFLGS    STATUS FLAGS
         1... ....  RCVTROFF    RACF HAS BEEN DEACTIVATED BY THE RVARY COMMAND
         .1.. ....  RCVTRDHD    RACF HAS BEEN RE-ACTIVATED BY RVARY AND REFRESH OF
                                THE RESIDENT ICB IS NECESSARY
         ..1. ....  RCVTSHR     THE RACF DATA SET AT SOME POINT DURING THIS IPL, WAS
                                ON A SHARED DASD DEVICE
         ...1 ....  RCVTNDUP    NO DUPLICATE DATA SET NAMES TO BE DEFINED
         .... 1...  RCVT24MD    AT LEAST ONE INSTALLATION EXIT HAS AMODE=24
         .... .1..  RCVTRMSG    RACF MESSAGE ICH412I WAS ISSUED
 154    (9A)     1  RCVTEROP    RACF TERMINAL OPTIONS
         1... ....  RCVTTERP    TERMINAL AUTHORIZATION CHECKING
```

OFFSETS		LENGTH	NAME	DESCRIPTION
		.1..	RCVTTUAC	DEFAULT UACC FOR TERMINALS NOT DEFINED TO RACF IF ON UACC = NONE IF OFF- UACC = READ
		..1.	RCVTAVIO	DO NOT CREATE LOG RECORD FOR COMMAND VIOLATIONS ONLY
		...1	RCVTSAUD	DO NOT AUDIT SPECIAL USER
		 1111		RESERVED
155	(9B)	1	RCVTPINV	GLOBAL MAX PASSWORD INTERVAL VALUE VALID RANGE 1-254
156	(9C)	4	RCVTRAU0	PTR TO AUDITING MODULE
160	(A0)	4	RCVTRIXP	PTR TO RACINIT POST PROCESSING INSTALLATION EXIT
164	(A4)	4	RCVTRCXP	PTR TO RACCHK POST PROCESSING INSTALLATION EXIT
168	(A8)	4	RCVTRID0	PTR TO MSC VERIFY RTN
172	(AC)	1	RCVTVERS	VERSION INDICATOR: 0 VERSION 1 RELEASE 1, 1 VERSION 1 RELEASE 2, 2 VERSION 1 RELEASE 3, 4 VERSION 1 RELEASE 4, 5 VERSION 1 RELEASE 5 6 VERSION 1 RELEASE 6 7 VERSION 1 RELEASE 7
173	(AD)	3	RCVTEXTA	RESERVED
176	(B0)	4	RCVTAPTR	ADDRESS FIELD RESERVED FOR APPLICATION USE
180	(B4)	4	RCVTNCX	PTR NAMING CONVENTION EXIT
184	(B8)	4	RCVTNCDX	PTR NAMING CONVENTION EXIT FOR DELETE FUNCTION
188	(BC)	4	RCVTCDTP	PTR TO CLASS DESC TABLE
192	(C0)	4	RCVTREXP	PTR TO RACSTAT MODULE
196	(C4)	4	RCVTFRCP	PTR TO FRACHECK MODULE
200	(C8)	4	RCVTFRXP	PTR TO FRACHECK EXIT
204	(CC)	4	RCVTRLX	PTR TO RACLIST PRE-EXIT
208	(D0)	4	RCVTRLXP	PTR TO RACLIST SELECTION EXIT
212	(D4)	4	RCVTCSTA	CLASS STATISTICS OPTION
216	(D8)	4	RCVTCAUD	CLASS AUDITING OPTIONS
220	(DC)	4	RCVTCPRO	CLASS PROTECTION OPTION
224	(E0)	4	RCVTDSDT	PTR TO DATA SET DESCRIPTOR TABLE

OFFSETS		LENGTH	NAME	DESCRIPTION
228	(E4)	4	RCVTRNGP	PTR TO RANGE TABLE
232	(E8)	4	RCVTAUTP	PTR TO RACF AUTHORIZED CALLER TABLE ICHAUTAB
236	(EC)	4	RCVTPWDX	PTR TO RACF PASSWORD EXIT.
240	(F0)	1	RCVTHIST	NUMBER OF PASSWORD GENERATIONS TO MAINTAIN AND CHECK AGAINST.
241	(F1)	1	RCVTRVOK	NUMBER OF CONSECUTIVE UNSUCCESSFUL ATTEMPTS BEFORE REVOKING A USERID.
242	(F2)	1	RCVTWARN	PASSWORD WARNING VALUE.
243	(F3)	1	RCVTINAC	INACTIVATE INTERVAL.
244	(F4)	80	RCVTSNTX	PASSWORD SYNTAX RULES.
244	(F4)	1	RCVTSLEN	STARTING LENGTH VALUE.
245	(F5)	1	RCVTELEN	ENDING LENGTH VALUE.
246	(F6)	8	RCVTRULS	CONTENT RULES.
246	(F6)	1	RCVTRUL1	CONTENT RULE.
247	(F7)	1	RCVTRUL2	CONTENT RULE.
248	(F8)	1	RCVTRUL3	CONTENT RULE.
249	(F9)	1	RCVTRUL4	CONTENT RULE.
250	(FA)	1	RCVTRUL5	CONTENT RULE.
251	(FB)	1	RCVTRUL6	CONTENT RULE.
252	(FC)	1	RCVTRUL7	CONTENT RULE.
253	(FD)	1	RCVTRUL8	CONTENT RULE.
324	(144)	4	RCVTMDEL	MODEL OPTIONS.
324	(144)	1		OPTIONS.
		1...	RCVTMGDG	MODEL-GDG IN EFFECT.
		.1..	RCVTMUSR	MODEL-USER IN EFFECT.
		..1.	RCVTMGRP	MODEL-GROUP IN EFFECT.
325	(145)	1		RESERVED.
326	(146)	1		RESERVED.
327	(147)	1		RESERVED.
328	(148)	1	RCVTWCNT	NUMBER OF VSL ENTRIES.
329	(149)	1	RCVTOPTX	OPTIONS.
		1...	RCVTLGRP	LIST-OF-GRPS CHKING ACTIVE.
330	(14A)	6		RESERVED.
336	(150)	32	RCVTVSL	VSL ENTRIES.
368	(170)	4	RCVTCGSN	NUMBER OF CONNECT-REMOVE COMMANDS ISSUED THAT ALTERED A USER'S AUTHORITY.

OFFSETS		LENGTH	NAME	DESCRIPTION
372	(174)	4	RCVTCGEN	CLASS MASK FOR GENERIC PROFILE CHECKING
376	(178)	4	RCVTCGCM	CLASS MASK FOR GENERIC COMMAND PROCESSING
380	(17C)	4	RCVTRDXP	PTR TO RACDEF POST PROCESSING INSTALLATION EXIT-ICHRDX02
384	(180)	4	RCVTFPB	BASE FOR FASTPATH TABLE.
388	(184)	4	RCVTFPTH	CLASS FASTPATH OPTIONS.
392	(188)	4	RCVTFLG1	MISC. OPTIONS.
		1...	RCVTFPDS	FASTPATH FOR DATASET CLASS
		.1..	RCVTTDSN	TAPE DATA SET PROTECTION IN EFFECT
		..11 1111		RESERVED.
		1...	RCVTPRO	PROTECT-ALL IN EFFECT
		.1..	RCVTPROF	1 PROTECT-ALL WARNING IN EFFECT 0 PROTECT-ALL FAILURE IN EFFECT (THIS FLAG IS IGNORED IF RCVTPRO HAS A VALUE OF'0'B)
		..1.	RCVTEOS	ERASE-ON-SCRATCH IN EFFECT
		...1	RCVTEOSL	ERASE-ON-SCRATCH BY SECLEVEL IN EFFECT (THIS FLAG IS IGNORED IF RCVTEOS HAS A VALUE OF'0'B)
		.... 1...	RCVTEOSA	ERASE-ON-SCRATCH FOR ALL DATASETS IN EFFECT (THIS FLAG IGNORED IF RCVTEOS HAS A VALUE OF '0'B)
		.... .111		RESERVED.
		1...	RCVTPROG	ACCESS CONTROL BY PROGRAM IN EFFECT
		.111 1111		
		1111 1111		RESERVED.
396	(18C)	2	RCVTRTPD	SYSTEM SECURITY RETENTION PERIOD
398	(18E)	1	RCVTSLVL	SECURITY LEVEL FOR ERASE-ON- SCRATCH
399	(18F)	1	RCVTQLLN	LENGTH OF SINGLE LEVEL DATASET NAME PREFIX
400	(190)	9	RCVTQUAL	INSTALLATION CONTROLLED PREFIX FOR SINGLE LEVEL DATASET NAMES, PLUS PERIOD FOR LEVEL
409	(199)	3		RESERVED.
412	(19C)	4	RCVTSPT	POINTER TO THE STARTED PROCEDURES TABLE (ICHRIN03)
416	(1A0)	4	RCVTDESX	POINTER TO THE PASSWORD ENCRYP- TION INSTALLATION EXIT (ICHDEX01)
420	(1A4)	4	RCVTNTAB	POINTER TO THE NAMING CONVENTION TABLE (ICHNCV00)
424	(1A8)	4	RCVTNRTN	POINTER TO THE NAMING CONVENTION ROUTINE (ICHNRT00)

OFFSETS		LENGTH	NAME	DESCRIPTION
428	(1AC)	4	RCVTFRX2	ADDRESS OF THE FRACHECK POST- PROCESSING INSTALLATION EXIT (ICHRFX02)
432	(1B0)	8	RCVTPROB	ADDRESSES OF CONTROLLED PROGRAMS LIST ANCHOR BLOCKS
432	(1B0)	4	RCVTCISP	ADDRESS OF CURRENT ANCHOR FOR CONTROLLED PROGRAMS LIST
436	(1B4)	4	RCVTOISP	ADDRESS OF OLD ANCHOR FOR CONTROLLED PROGRAMS LIST
440	(1B8)	8	RCVTSWPW	PASSWORD FOR RVARY SWITCH
448	(1C0)	8	RCVTINPW	PASSWORD FOR RVARY STATUS
456	(1C8)	4	RCVTLARP	PTR TO LINKAGE ASSIST ROUTINE FOR INSTAL EXITS (ICHLAR00)
460	(1CC)	4	RCVTCTV0	ADDRESS OF TVTOC UTILITY (ICHCTV00)
464	(1D0)	48		RESERVED.
512	(200)	0		END OF RCVT

	OFFSETS /		OFFSETS /		OFFSETS /
NAME	EQU VALUE	NAME	EQU VALUE	NAME	EQU VALUE
				RCVT	0 (0)
RCVTADAS	151 X'08'	RCVTADAT	151 X'10'	RCVTAGRO	151 X'40'
RCVTAOPR	151 X'01'	RCVTAPTR	176 (B0)	RCVTATAP	151 X'04'
RCVTATER	151 X'02'	RCVTAUOP	151 (97)	RCVTAUSE	151 X'20'
RCVTAUTP	232 (E8)	RCVTAVIO	154 X'20'	RCVTAXTA	152 (98)
RCVTBAM	44 (2C)	RCVTCAUD	216 (D8)	RCVTCDTP	188 (BC)
RCVTCGCM	376(178)	RCVTCGEN	372(174)	RCVTCGSN	368(170)
RCVTCISP	432(1B0)	RCVTCPRO	220 (DC)	RCVTCSTA	212 (D4)
RCVTCTVO	460(1CC)	RCVTDASD	150 X'40'	RCVTDCB	4 (4)
RCVTDEB	8 (8)	RCVTDESX	416(1A0)	RCVTDGCM	150 X'10'
RCVTDGEN	150 X'20'	RCVTDSDT	224 (E0)	RCVTDSN	56 (38)
RCVTDSNL	52 (34)	RCVTELEN	245 (F5)	RCVTEOS	393 X'20'
RCVTEOSA	393 X'08'	RCVTEOSL	393 X'10'	RCVTEROP	154 (9A)
RCVTEXTA	173 (AD)	RCVTFLGS	153 (99)	RCVTFLG1	392(188)
RCVTFPB	384(180)	RCVTFPDS	392 X'80'	RCVTFPTH	388(184)
RCVTFRCP	196 (C4)	RCVTFRXP	200 (C8)	RCVTFRX2	428(1AC)
RCVTHDR	20 (14)	RCVTHIST	240 (F0)	RCVTID	0 (0)
RCVTINAC	243 (F3)	RCVTINDX	12 (C)	RCVTINPW	448(1C0)
RCVTISTL	48 (30)	RCVTJALL	150 X'01'	RCVTJCHK	150 X'02'
RCVTJXAL	150 X'04'	RCVTLARP	456(1C8)	RCVTLGRP	329 X'80'
RCVTMDEL	324(144)	RCVTMGDG	324 X'80'	RCVTMGRP	324 X'20'
RCVTMUSR	324 X'40'	RCVTNADS	53 X'02'	RCVTNCDX	184 (B8)
RCVTNCX	180 (B4)	RCVTNDSS	53 X'20'	RCVTNDUP	153 X'10'
RCVTNDVS	53 X'08'	RCVTNLS	53 X'40'	RCVTNREC	54 (36)
RCVTNRTN	424(1A8)	RCVTNTAB	420(1A4)	RCVTNTMS	53 X'04'
RCVTNTVS	53 X'10'	RCVTOISP	436(1B4)	RCVTOPTX	329(149)
RCVTPINV	155 (9B)	RCVTPRO	393 X'80'	RCVTPROB	432(1B0)
RCVTPROF	393 X'40'	RCVTPROG	394 X'80'	RCVTPWDX	236 (EC)
RCVTQLLN	399(18F)	RCVTQUAL	400(190)	RCVTRAUO	156 (9C)
RCVTRCX	28 (1C)	RCVTRCXP	164 (A4)	RCVTRDHD	153 X'40'
RCVTRDSN	150 X'08'	RCVTRDX	32 (20)	RCVTRDXP	380(17C)
RCVTREXP	192 (C0)	RCVTRIDO	168 (A8)	RCVTRIX	24 (18)
RCVTRIXP	160 (A0)	RCVTRLX	204 (CC)	RCVTRLXP	208 (D0)
RCVTRMSG	153 X'04'	RCVTRNA	53 X'80'	RCVTRNGP	228 (E4)
RCVTROFF	153 X'80'	RCVTRTPD	396(18C)	RCVTRUCB	36 (24)
RCVTRULS	246 (F6)	RCVTRUL1	246 (F6)	RCVTRUL2	247 (F7)
RCVTRUL3	248 (F8)	RCVTRUL4	249 (F9)	RCVTRUL5	250 (FA)
RCVTRUL6	251 (FB)	RCVTRUL7	252 (FC)	RCVTRUL8	253 (FD)
RCVTRVOK	241 (F1)	RCVTSAUD	154 X'10'	RCVTSHR	153 X'20'
RCVTSLEN	244 (F4)	RCVTSLVL	398(18E)	RCVTSNTX	244 (F4)
RCVTSPT	412(19C)	RCVTSTAT	53 (35)	RCVTSTA1	150 (96)
RCVTSWPW	440(1B8)	RCVTTAPE	150 X'80'	RCVTTDSN	392 X'40'
RCVTTEMP	16 (10)	RCVTTERP	154 X'80'	RCVTTUAC	154 X'40'
RCVTUADS	100 (64)	RCVTUVOL	144 (90)	RCVTVERS	172 (AC)
RCVTVSL	336(150)	RCVTWARN	242 (F2)	RCVTWCNT	328(148)
RCVTXLEN	40 (28)	RCVT24MD	153 X'08'	END OF	RCVT

RNG

Common Name: RACF Data Set Range Table
Macro ID: ICHPRNG
DSECT Name: ICHPRNG
Created by: Installation provided
Subpool and Key: 252 and key 0
Size: 44 byte header plus 45 bytes/entry
Pointed to by: RCVTRNGP field of the PCVT data area.
Serialization: None
Function: This table describes the alphabetic range of profiles contained in each RACF data set.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	4	ICHPRNG	RANGE TABLE MAPPING
0	(0)	4	RNGNUM	NUMBER OF ARRAY ELEMENTS
4	(4)	0	RNGVALS	ARRAY OF RANGE/DS-NUMBER PAIRS
4	(4)	44	RNGSTART	LOWER BOUND OF RANGE
48	(30)	1	RNGDSNUM	DATA SET SEQUENCE NUMBER

RRPF

Common Name: Resident Profile Map
Macro ID: ICHRRPF
DSECT Name: RRPF,DSPVOLS,DSPACCES,DSPINSTD,DSPDPTD,DSP2ACCS
Created by: RACHECK SVC
Subpool and Key: Subpool 231 and key 0 when CSA profile requested;
Subpool 229 and key 0 when private profile requested
Size: Variable
Pointed to by: ACEEAMP field of the ACEE data area, or returned in Register 1 after RACHECK request.
Serialization: None
Function: This area maps a profile for general resource used for authorization checking, and RACDEF modeling function.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	120	RRPF	RESIDENT PROFILE MAP
0	(0)	4	DSPCORE	
0	(0)	1	RRPSP	AREA SUBPOOL NUMBER
1	(1)	3	RRPLEN	TOTAL AREA LENGTH
4	(4)	116	RRPVDATA	PROFILE DATA
4	(4)	116	DSPSUB	
4	(4)	44	DSPDSNM	RESOURCE NAME IF CSA PROFILE REQUESTED, OR PROFILE NAME IF PRIVATE AREA PROFILE REQUESTED
48	(30)	1	DSPUACC	UNIVERSAL ACCESS
49	(31)	1	DSPAUDIT	AUDIT FLAGS
50	(32)	1	DSPTYPE	D.S. TYPE FLAGS
		1...	DSPTP	1 VSAM, 0 NON-VSAM
		.1..	DSPMDL	1 MODEL.
		..1.	DSPTAPE	1 TAPE.
		...1 1111		RESERVED
51	(33)	1	DSPLEVEL	RESOURCE LEVEL
52	(34)	4	DSPVOLOF	OFFSET TO VOLSER LIST
56	(38)	4	DSPACCOF	OFFSET TO ACCESS LIST
60	(3C)	8	DSPCLASS	RESOURCE CLASS
68	(44)	1	DSPGAUD	GLOBAL AUDIT FLAG
69	(45)	1	DSPVRSN	VERSION = 0
70	(46)	1	DSPWARN	WARNING FLAG BIT 7 = 1 RESOURCE HAS WARNING ATTRIBUTE
71	(47)	1	DSPEOS	ERASE-ON-SCRATCH FLAG BIT 0 = 1 DATASET WILL BE ERASED WHEN SCRATCHED

OFFSETS		LENGTH	NAME	DESCRIPTION
72	(48)	4	DSPINST	OFFSET TO INSTALLATION DATA
76	(4C)	4	DSPNEXTP	ADDR NEXT MODEL
80	(50)	1	DSPFNF	MODEL FOUND INDICATOR 0,FD 1,NFD
81	(51)	1	DSPSLVL	RESOURCE SECURITY LEVEL
82	(52)	2	DSPRTPD	RETENTION PERIOD
84	(54)	8	DSPOWNER	RESOURCE OWNER
92	(5C)	8	DSPNOTFY	USERID TO NOTIFY WHEN THIS PROFILE DENIES ACCESS
100	(64)	4	DSPDPTOF	OFFSET TO CATEGORY LIST
104	(68)	4	DSPPGMOF	OFFSET TO CONDITIONAL ACCESS LIST
108	(6C)	1	DSPRESF	RESOURCE FLAG (ONLY FOR TAPE VOLUMES BIT 0 = 1 VOLUME MAY ONLY CONTAIN ONE DATA SET BIT 2 = 1 VOLUME MAY NOT CONTAIN A TVTOC)
109	(6D)	1	DSPTDAYS	DAYS THAT THE TERMINAL MAY NOT BE USED (BIT 0 SUNDAY, BIT 1 MONDAY,...
110	(6E)	3	DSPLOGNT	EARLIEST TIME THAT THE TERMINAL BE USED.(HHMM)
113	(71)	3	DSPLOGFT	LATEST TIME THAT THE TERMINAL BE USED.(HHMM)
116	(74)	3	DSPTZONE	TIME OFFSET OF TERMINAL FROM THE CPU. (+ = EAST, = WEST)
119	(77)	1		RESERVED
0	(0)	2	DSPVOLS	VOLSER LIST
0	(0)	2	DSPVOLCT	NUMBER OF ENTRIES
2	(2)	0	DSPVOLSR	VOLSERS
0	(0)	2	DSPACCES	ACCESS LIST
0	(0)	2	DSPACT	NUMBER OF ENTRIES
2	(2)	0	DSPACCLE	ACCESS LIST ENTRIES
2	(2)	8	DSPAUSER	USERID/GRPNAME
10	(A)	1	DSPACS	ACCESS AUTHORITY
0	(0)	2	DSPINSTD	INSTALLATION DATA
0	(0)	2	DSPLINST	LENGTH OF INSTALLATION DATA
2	(2)	0	DSPIDATA	INSTALLATION DATA
0	(0)	2	DSPDPTD	CATEGORY LIST
0	(0)	2	DSPDPTCT	NUMBER OF CATEGORIES
2	(2)	0	DSPDEPT	CATEGORY LIST

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	2	DSP2ACCS	SECOND ACCESS LIST
0	(0)	2	DSP2GCT	NUMBER OF ENTRIES
2	(2)	0	DSP2ACCL	ACCESS LIST ENTRIES
2	(2)	8	DSP2ENT	ENTITY NAME
10	(A)	8	DSP2USR	USER/GROUP ID
18	(12)	1	DSP2ACS	ACCESS AUTHORITY

TEMPTAB

Common Name: RACF Resident Template Table
Macro ID: ICHPTEMP
DSECT Name: TEMPTAB
Created by: ICHSEC00
Subpool and Key: 241 and key 0
Size: variable
Pointed to by: RCVTTEMP
Serialization: None
Function: This control block contains pointers to all resident templates.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	44	TEMPTAB	LOCATED FROM RACF CVT
0	(0)	4		RESERVED
4	(4)	40	TEMPPTR	AN ARRAY OF POINTERS TO RESIDENT TEMPLATES THIS ARRAY IS INDEXED BY TEMPLATE NUMBER TEMPPTR(N) WILL BE ZERO IF THAT TEMPLATE DOES NOT EXIST

XTAB

Common Name: RACF Resident Index Block Table
Macro ID: ICHPXTAB
DSECT Name: XTAB
Created by: ICHSEC00
Subpool and Key: 241 and key 0
Size: 128 bytes
Pointed to by: RCVTINDX
Serialization: None
Function: This control block contains data set locators (RBA) and storage pointers for each of the resident RACF data set index blocks.

OFFSETS		LENGTH	NAME	DESCRIPTION
0	(0)	8	XTAB	LOCATED FROM RACF CVT
0	(0)	8	XTABHDR	NOTE: THIS STRUCTURE MUST BE KEPT ALIGNED SUCH THAT THE INCORE POOL LOCK, XTABBLOC, IS ON A DOUBLEWORD BOUNDARY. ICHSEC00 MUST ALLOCATE STORAGE FOR XTAB IN SUCH A WAY THAT THE ADDRESS IN RCVTINDX IS DOUBLEWORD ALIGNED.
0	(0)	3		RESERVED
3	(3)	1	XTABNUM	NUMBER OF RESIDENT RACF BLOCKS
4	(4)	4	XTABHREF	POOL REFERENCE COUNT
8	(8)	0	XTABARAY	ARRAY OF ADDRESSES AND RBA LOCATORS FOR RESIDENT BLOCKS
8	(8)	4	XTABPTR	STORAGE PTR OF BLOCK
12	(C)	6	XTABRBA	RBA LOCATOR OF BLOCK
18	(12)	1	XTABNDXL	BLOCK LEVEL NUMBER
19	(13)	1	XTABSFLG	BLOCK ENTRY STATUS
	1...		XTABRFSH	ON- FOR SHARED RACF DATA SET THIS RESIDENT BLOCK MUST BE REFRESHED BEFORE USE
20	(14)	4	XTABCHCT	CHANGE COUNT
24	(18)	8	XTABBLOC	LOCK ON BLOCK- MUST BE DOUBLEWORD ALIGNED. THIS FIELD IS ZERO IF THE BLOCK IS UNLOCKED, AND CONTAINS THE LOCKING TASK'S TCB AND ASID OTHERWISE.
32	(20)	4	XTABBREF	BLOCK REFERENCE COUNT
36	(24)	4		RESERVED

Index

Cut or Fold Along Line

System Programming
Library:
Resource Access Control
Facility (RACF)

SC28-1343-1

READER'S
COMMENT
FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation? ____

How do you use this publication? ____________________________________

Number of latest Newsletter associated with this publication: ____________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

System Programming Library: Resource Access Control Facility (RACF)

SC28-1343-1 S370-34

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

System Programming Library: Resource Access Control Facility (RACF)

SC28-1343-1

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity   Accuracy   Completeness   Organization   Coding   Retrieval   Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation? ______________________________

How do you use this publication? ______________________________

Number of latest Newsletter associated with this publication: ______________________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Cut or Fold Along Line

System Programming Library: Resource Access Control Facility (RACF)

SC28-1343-1 S370-34

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

System Programming
Library:
Resource Access Control
Facility (RACF)

SC28-1343-1

READER'S
COMMENT
FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

Cut or Fold Along Line

What is your occupation? ________________________________

How do you use this publication? ____________________________

Number of latest Newsletter associated with this publication: ________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

System Programming Library: Resource Access Control Facility (RACF)

SC28-1343-1 S370-34

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

System Programming Library: Resource Access Control Facility (RACF)

SC28-1343-1

S370-34

Printed in U.S.A.

SC28-1343-01