z/VM
7.3

Security Server
RACROUTE Macro Reference

SC24-6324-73

Note:

Before you use this information and the product it supports, read the information in “Notices” on page 445.

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2022-09-07

© Copyright International Business Machines Corporation 1990, 2022.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures vii

Tables ix

About This Document xi
Intended Audience xi
Where to Find More Information xi
Links to Other Documents and Websites xi

How to Send Your Comments to IBM xiii

Summary of Changes for z/VM: Security Server RACROUTE Macro Reference xv
SC24-6324-73, z/VM 7.3 (September 2022) xv
SC24-6324-02 z/VM Version 7.2 (July 2021) xv
SC24-6324-02 z/VM Version 7.2 (September 2020) xv
SC24-6324-01, z/VM 7.1 (May 2020) xv
Multi-Factor Authentication for z/VM xv
SC24-6324-00, z/VM 7.1 (September 2018) xv

Chapter 1. How to Use the RACF System Macros 1
Reading the Macro Instructions 3
Continuation Lines 5

Chapter 2. RACF System Macros 7
RACROUTE: Router Interface 8
Keyword and Parameter Cross-Reference for RACROUTE 9
Addressing Considerations 11
Special Considerations for Using RACROUTE on z/VM 12
Authorization to Issue RACROUTE Requests 12
Issuing RACROUTE Requests on CMS 13
Issuing RACROUTE Requests on GCS 13
Event Control Blocks (ECBs) and Their Significance on z/VM 14
ACIGROUP Considerations for the ENTITY Parameter 15
RACROUTE (Standard Form) 16
RACROUTE (List Form) 23
RACROUTE (Execute Form) 24
RACROUTE (Modify Form) 26
RACROUTE REQUEST=AUDIT: General-Purpose Security-Audit Request 27
RACROUTE REQUEST=AUDIT (Standard Form) 28
RACROUTE REQUEST=AUDIT (List Form) 32
RACROUTE REQUEST=AUDIT (Execute Form) 33
RACROUTE REQUEST=AUDIT (Modify Form) 34
RACROUTE REQUEST=AUTH: Check RACF Authorization 35
RACROUTE REQUEST=AUTH (Standard Form) 35
RACROUTE REQUEST=AUTH (List Form) 47
RACROUTE REQUEST=AUTH (Execute Form) 50
RACROUTE REQUEST=AUTH (Modify Form) 53
RACROUTE REQUEST=DEFINE: Define, Modify, Rename, or Delete a Resource for RACF 55
RACROUTE REQUEST=DEFINE (Standard Form) 56

RACROUTE REQUEST=DEFINE (List Form)....74
RACROUTE REQUEST=DEFINE (Execute Form).... 78
RACROUTE REQUEST=DEFINE (Modify Form).... 82
RACROUTE REQUEST=DIRAUTH: Check RACF-Directed Authorization to a Sent Message.... 86
RACROUTE REQUEST=DIRAUTH (Standard Form)....86
RACROUTE REQUEST=DIRAUTH (List Form)....89
RACROUTE REQUEST=DIRAUTH (Execute Form).... 90
RACROUTE REQUEST=DIRAUTH (Modify Form)....91
RACROUTE REQUEST=EXTRACT: Replace or Retrieve Fields.... 92
RACROUTE REQUEST=EXTRACT (Standard Form).... 93
RACROUTE REQUEST=EXTRACT (List Form).... 113
RACROUTE REQUEST=EXTRACT (Execute Form)....116
RACROUTE REQUEST=EXTRACT (Modify Form)....118
RACROUTE REQUEST=FASTAUTH: Verify Access to Resources.... 121
RACROUTE REQUEST=FASTAUTH (Standard Form)....121
RACROUTE REQUEST=FASTAUTH (List Form)....125
RACROUTE REQUEST=FASTAUTH (Execute Form).... 126
RACROUTE REQUEST=LIST: Build In-Storage Profiles.... 128
RACROUTE REQUEST=LIST (Standard Form)....128
RACROUTE REQUEST=LIST (List Form).... 134
RACROUTE REQUEST=LIST (Execute Form)....135
RACROUTE REQUEST=LIST (Modify Form)....137
RACROUTE REQUEST=STAT: Determine RACF Status.... 138
RACROUTE REQUEST=STAT (Standard Form).... 138
RACROUTE REQUEST=STAT (List Form).... 141
RACROUTE REQUEST=STAT (Execute Form).... 141
RACROUTE REQUEST=STAT (Modify Form).... 143
RACROUTE REQUEST=TOKENBLD: Build a UTOKEN.... 144
RACROUTE REQUEST=TOKENBLD (Standard Form).... 144
RACROUTE REQUEST=TOKENBLD (List Form).... 148
RACROUTE REQUEST=TOKENBLD (Execute Form)....150
RACROUTE REQUEST=TOKENBLD (Modify Form)....152
RACROUTE REQUEST=TOKENMAP: Access Token Fields....153
RACROUTE REQUEST=TOKENMAP (Standard Form).... 153
RACROUTE REQUEST=TOKENMAP (List Form).... 155
RACROUTE REQUEST=TOKENMAP (Execute Form)....156
RACROUTE REQUEST=TOKENMAP (Modify Form)....157
RACROUTE REQUEST=TOKENXTR: Extract UTOKENS....158
RACROUTE REQUEST=TOKENXTR (Standard Form)....158
RACROUTE REQUEST=TOKENXTR (List Form).... 160
RACROUTE REQUEST=TOKENXTR (Execute Form)....161
RACROUTE REQUEST=TOKENXTR (Modify Form)....162
RACROUTE REQUEST=VERIFY: Identify and Verify a RACF-Defined User.... 162
RACROUTE REQUEST=VERIFY (Standard Form)....163
RACROUTE REQUEST=VERIFY (List Form).... 176
RACROUTE REQUEST=VERIFY (Execute Form)....179
RACROUTE REQUEST=VERIFY (Modify Form)....182
RACROUTE REQUEST=VERIFYX: Verify User and Return a UTOKEN.... 185
RACROUTE REQUEST=VERIFYX (Standard Form)....185
RACROUTE REQUEST=VERIFYX (List Form)....196
RACROUTE REQUEST=VERIFYX (Execute Form).... 199
RACROUTE REQUEST=VERIFYX (Modify Form).... 201
RACSYNC Macro (z/VM Only).... 204

Appendix A. Independent RACF System Macros..... 207
FRACHECK Macro.... 208
FRACHECK (Standard Form).... 208

FRACHECK (List Form)....211
FRACHECK (Execute Form)....213
RACDEF: Define a Resource to RACF....214
RACDEF (Standard Form)....214
RACDEF (List Form)....228
RACDEF (Execute Form)....231
RACHECK: Check RACF Authorization....234
RACHECK (Standard Form)....235
RACHECK (List Form)....245
RACHECK (Execute Form)....247
RACINIT: Identify a RACF-Defined User....249
RACINIT (Standard Form)....249
RACINIT (List Form)....257
RACINIT (Execute Form)....259
RACLIST: Build In-Storage Profiles....261
RACLIST (Standard Form)....261
RACLIST (List Form)....265
RACLIST (Execute Form)....266
Limited Function RACROUTE on z/VM (RACROUTE REQUEST=AUTH with RELEASE=1.8.2 specified)....267
RACROUTE on z/VM (Standard Form)....268
RACROUTE on z/VM (List Form)....271
RACROUTE on z/VM (Execute Form)....272
RACROUTE: SAF Router Interface....273
RACSTAT: RACF Status....273
RACSTAT (Standard Form)....273
RACSTAT (List Form)....275
RACSTAT (Execute Form)....276
RACXTRT Macro....277
RACXTRT (List Form)....288
RACXTRT (Execute Form)....290

Appendix B. RACF Database Templates....293

Appendix C. RACROUTE Interface to an External Security Manager Product (Non-RACF) on z/VM....337
Providing RPIUCMS Module....337
Providing RPIATGCS Module and RPIGCS LOADLIB....338
Providing RACROUTE Support Code....339
Entry....340
Exit....341

Appendix D. Requesting Security Services....343

Appendix E. Sample RACROUTE Program for Shared User IDs....345

Appendix F. Data Areas for RACROUTE....347
ACEE....347
Constants....354
Cross Reference....354
ACHKL....356
Cross Reference....360
AUL....362
Cross Reference....363
CGRP....364
Constants....366

Cross Reference..... 366
DAUT..... 367
Cross Reference..... 367
FAST..... 368
Cross Reference..... 369
ISP..... 369
Constants..... 375
Cross Reference..... 375
RCVT..... 378
Constants..... 391
Cross Reference..... 391
RDDFL..... 397
Cross Reference..... 401
RIPL..... 404
Cross Reference..... 409
RLST..... 412
Cross Reference..... 413
RRPF..... 414
Constants..... 418
Cross Reference..... 418
RUTKN..... 421
Constants..... 423
Cross Reference..... 424
RXTL..... 425
Constants..... 428
Cross Reference..... 428
RXTW..... 429
Cross Reference..... 432
SAFP..... 434
Constants..... 436
Cross Reference..... 439
SAFV..... 440
Constants..... 441
Cross Reference..... 441
STAT..... 441
Cross Reference..... 442
TSRV..... 442
Cross Reference..... 443

Notices..... 445
Programming Interface Information..... 446
Trademarks..... 446
Terms and Conditions for Product Documentation..... 446
IBM Online Privacy Statement..... 447

Bibliography..... 449
Where to Get z/VM Information..... 449
z/VM Base Library..... 449
z/VM Facilities and Features..... 450
Prerequisite Products..... 452
Related Products..... 452

Index..... 455

Figures

1. Sample Macro Instruction.....4

2. Continuation Coding.....5

Tables

1. RACROUTE REQUEST=type and Independent RACF System Macros..... 8

2. RACROUTE REQUEST=keyword Cross-Reference.....9

3. Types of Profile Checking Performed by RACROUTE REQUEST=AUTH.....42

4. Cross-reference for RACROUTE REQUEST=type and the Independent RACF System Macros..... 207

5. FRACHECK Parameters for RELEASE=1.6 through 1.8.1..... 210

6. RACDEF Parameters for RELEASE= 1.6 through 1.8.1..... 224

7. Types of Profile Checking Performed by RACHECK..... 240

8. RACHECK Parameters for RELEASE=1.6 through 1.8.2.....242

9. RACINIT Parameters for RELEASE=1.6 through 1.8.1..... 254

10. RACLIST Parameters for RELEASE=1.6 through 1.8.1..... 263

11. RACSTAT Parameters for RELEASE=1.6 through 1.8.1..... 274

12. RACXTRT Parameters for RELEASE=1.6 through 1.8.1..... 286

13. RACROUTE Macro Keywords and Their Request-Specific Parameter Lists..... 340

14. Return Codes for Register 15 and the SAFPSFRC Field in the RACROUTE Parameter List..... 341

About This Document

This document contains information on how to use the system macro instructions provided with the IBM RACF® Security Server for z/VM.

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

Intended Audience

This publication is intended to be used by programmers who are writing applications that need to invoke RACF (or another external security product). It is also written for programmers who write other external security products (that replace RACF) to perform the following functions:

- Centralized auditing
- Resource authorization
- Resource definition
- Data encryption
- User identification and verification.

Where to Find More Information

For information about related publications, refer to the “Bibliography” on page 449.

Links to Other Documents and Websites

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

© Copyright IBM Corp. 1990, 2022

How to Send Your Comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

To send us your comments, go to z/VM Reader's Comment Form (https://www.ibm.com/systems/campaignmail/z/zvm/zvm-comments) and complete the form.

If You Have a Technical Problem

Do not use the feedback method. Instead, do one of the following:

- Contact your IBM service representative.
- Contact IBM technical support.
- See IBM: z/VM Support Resources (https://www.ibm.com/vm/service).
- Go to IBM Support Portal (https://www.ibm.com/support/entry/portal/Overview).

© Copyright IBM Corp. 1990, 2022

Summary of Changes for z/VM: Security Server RACROUTE Macro Reference

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

SC24-6324-73, z/VM 7.3 (September 2022)

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

SC24-6324-02 z/VM Version 7.2 (July 2021)

This edition includes terminology, maintenance, and editorial changes.

SC24-6324-02 z/VM Version 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

SC24-6324-01, z/VM 7.1 (May 2020)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.1.

Multi-Factor Authentication for z/VM

With the PTF for APAR VM66338, Multi-Factor Authentication (MFA) provides for the establishment of a user's identity by utilizing more than one type of authentication. This provides greater security by allowing for an additional form of proof in the event that one token (for example, a password) becomes compromised. Previously, authentication of identity during the logon process could be met only by using a password or passphrase. MFA enables support for an external service to authenticate tokens that have been generated after a successful multi-factor authentication.

SC24-6324-00, z/VM 7.1 (September 2018)

This edition supports the general availability of z/VM 7.1.

© Copyright IBM Corp. 1990, 2022

Chapter 1. How to Use the RACF System Macros

There are four different forms of the RACF macros: standard (S), list (L), execute (E), and modify (M). An explanation of when and why to use each form follows.

- Standard form (MF=S):

  Use the standard form of the macro when writing your own user programs (such as a nonsystem module), programs that you store in your own loadlib. By design, the standard form of the macro generates an inline parameter list and then modifies it. Do not use the standard form of the macro to write reentrant code, because reentrant code cannot be modified. With few exceptions, if you use the standard form of the macro in writing reentrant code, the execution of the code results in an abend.

  The standard form of the macro does three things: It obtains storage, fills in the parameters you have specified on the parameter list, and generates a call to the service routine.

- List, execute, and modify forms:

  Use the list, execute, and modify forms of the macro in combination when you write a reentrant program or plan to have numerous invocations of the macro. The forms of the macro work together in the following way:

  – List form (MF=L):

    You can use the list form in two ways:

    1. Allocate storage in your program's dynamic area (DSECT), and
    2. Provide a template in the control section (CSECT) from which the dynamic storage parameter list can be initialized.

    This implies that two list forms are generally used in one program. One is used to allocate storage, and the other is used to initialize that storage. For example, the parameter list length is copied from the control section parameter list to the dynamic storage parameter list. To ensure that a valid dynamic storage parameter list has been built, the entire parameter list residing in the control section should then be copied to the parameter list residing in the dynamic storage.

    Since parameter list lengths can change from one release of RACF to the next, it is important to specify the same release on all invocations of the macro whether they be list, modify, or execute forms. If you were to code RELEASE=1.9.2 on the control section parameter list and RELEASE=1.8 on the dynamic storage parameter list, the copy could result in an abend, or the call to the service would yield unpredictable results, since the complete parameter list was not copied.

    Note: The expansion of the list form does not contain any executable instructions; therefore, you cannot use registers in the list form.

  – Execute form (MF=E):

    When you specify the execute form of the macro, you can change the initial parameters you specified on the list form of the macro. You can also specify additional allowable parameters you may not have specified on the list form. When you issue the execute form of the macro, you generate a call to the service routine. You can change the parameters on the macro with each subsequent invocation of the execute form of the macro.

  – Modify form (MF=M):

    When you specify the modify form of the macro, you, in effect modify the parameter list of the list form of the macro. When you set the parameters that you want using the modify form of the macro, you can then use the execute form. The advantage of using the modify form is that it allows you to set only those parameters that you need. Thus, you can code a series of modify forms, followed by one execute form instead of many execute forms. This results in reducing the number of macro invocations that you need to code.

© Copyright IBM Corp. 1990, 2022

Note: You must use the modify form of the macro in conjunction with the execute or modify forms. The list form initializes certain fields that the execute and modify forms will not modify. Also, you must be sure to specify the same values for RELEASE= and REQUEST= on the execute list, and modify forms.

Following is a representation of the relationship between the list and execute forms of the RACROUTE macro.

```
*********************************************************************
*
* Example of list and execute in a reentrant module.
*
*********************************************************************
*
RACROUT  START
*        .
*        .
         BALR 12,0
         USING *,12
*
         USING DYNDAT,13
*
*********************************************************************
```

```
*********************************************************************
*
* Copy the static RACROUTE parameter list to the dynamic storage
* parameter list.
*
*********************************************************************
*
         LA  8,RACROUD                Load the address of the
*                                     dynamic storage parameter list.
*
         LA 10,RACROUS                Load the address of the static
*                                     parameter list.
*
         L   9,RACROUL                Load the length of the
*                                     parameter list.
*
         LR 11,9                      Copy the length into register 11
*                                     for the MVCL.
*
         MVCL 8,10                    Copy the static parameter list
*                                     into the dynamically allocated
*                                     storage.
*
*********************************************************************
*
* Establish addressability to RACROUTE parameters.
*
*********************************************************************
*
         LA 3,TOKNOUT
         USING TOKEN,3
         MVI TOKLEN,TOKCURLN          Initialize the TOKNOUT area
*                                     with the length.
*
         MVI TOKVERS,TOKVER01         Initialize the TOKNOUT area
                                      with the version.
*
         LA 4,USERLN                  Load register 4 with the
*                                     user ID information address.
*
         LA 5,SAFWK                   Load register 5 with the SAF
*                                     work area address.
*
RACRTE   RACROUTE REQUEST=VERIFYX,TOKNOUT=(3),SESSION=RJEBATCH,
               USERID=(4),PASSWRD=PASSLN,
               WORKA=(5),MF=(E,RACROUD),RELEASE=1.9
*
*********************************************************************
```

```
*********************************************************************
*
* Constants for RACROUTE
*
*********************************************************************
*
```

```
USERLN   DC    X'07'                  Length of user ID
USERID   DC    CL8'IBMUSER '          User ID value
PASSLN   DC    X'03'                  Password length
PASSWD   DC    CL8'IBM'               Password value
         DS OF
RACROUS  RACROUTE REQUEST=VERIFYX,MF=L,RELEASE=1.9
RACROUL  DC A(*-RACROUS)
*
         ICHSAFP                      SAF parameter list
*
&TOKCNST; SETB 1                       Allow additional constants
         ICHRUTKN                     Security TOKEN
*
*********************************************************************
*
* Module acquired dynamic storage.
*
*********************************************************************
*
DYNDAT   DSECT
*        .
*        .
*        .
RACROUD  RACROUTE REQUEST=VERIFYX,MF=L,RELEASE=1.9
                                      Acquire storage for the parameter
                                      list in the module dynamic storage
                                      area.
SAFWK    DS    128F'0'                SAF work area
TOKNOUT  DS    20F'0'                 Storage for TOKEN to be returned
*
         END   RACROUT
*
*********************************************************************
```

RACF macros are assembler macros; therefore you must invoke them in assembler statements. When you code a macro instruction, the assembler processes it by using the macro definitions supplied by IBM and placed in the macro library when the system is generated.

The assembler expands the macro instruction into executable machine instructions or data fields that are in the form of assembler-language statements, or both. The machine instructions branch around the data fields, load registers, and give control to the system. The instruction that gives control to the system for RACROUTE is a branch instruction. The macro expansion appears as part of the assembler output listing.

Note: High-level Assembler Release 4, or an equivalent assembler, is now required to assemble the RACROUTE macros.

The data fields, which are derived from parameters of the macro instruction, are used at execution time by the control program routine that performs the service associated with the macro.

Reading the Macro Instructions

Each macro description begins with a syntax diagram.

The syntax layout assumes that the standard begin, end, and continue columns are used. Therefore, column 1 is assumed to be the begin column. To change the begin, end, and continue columns, use the ICTL instruction to establish the coding format you want to use. If you do not use ICTL, the assembler recognizes the standard columns. To code the ICTL instruction, see the *High Level Assembler for MVS & z/VM & VSE Language Reference*.

Figure 1 on page 4 shows a sample macro instruction, RACROUTE REQUEST=AUTH, and summarizes all the coding information that is available for it. The example is divided into three columns, A, B, and C.

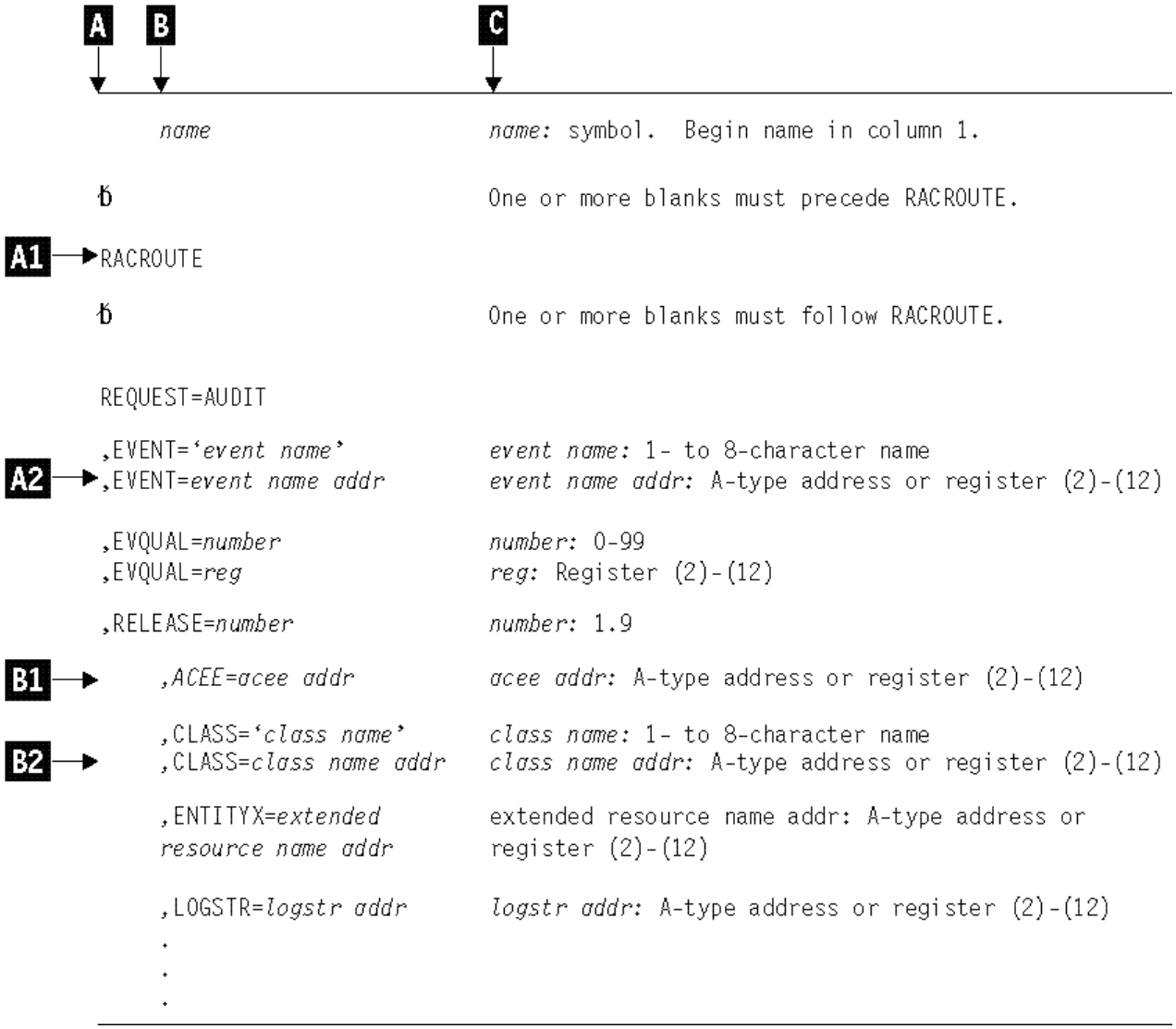


Figure 1. Sample Macro Instruction

- The first column, A, contains those parameters that are required for that macro instruction. If a single line appears in that column, A1, the parameter on that line is required and you must code it. If two or more lines appear together, A2, you must code the parameter appearing on one and only one of the lines.
- The second column, B, contains those parameters that are optional for that macro instruction. If a single line appears in that column, B1, the parameter on that line is optional. If two or more lines appear together, B2, the entire parameter is optional, but if you elect to make an entry, code one and only one of the lines.
- The third column, C, provides additional information about coding the macro instruction.

When substitution of a variable is required in column C, the following classifications are used:

symbol
Any symbol valid in the assembler language. That is, an alphabetic character followed by 0-7 alphameric characters, with no special characters and no blanks.

Rx-type address
Any address that is valid in an Rx-type instruction (such as LA).

register (2)-(12)
One of general registers 2 through 12, specified within parentheses, previously loaded with the right-adjusted value or address indicated in the parameter description. You must set the unused high-order bits to zero. You can designate the register symbolically or with an absolute expression.

decimal digit
Any decimal digit up to the value indicated in the parameter description. If both symbol and decimal digit are indicated, an absolute expression is also allowed.

register (1)
General register 1, previously loaded with the right-adjusted value or address indicated in the parameter description. You must set the unused high-order bits to zero. Designate the register as (1) only.

A-type address
Any address that can be written in an A-type address constant.

default
A value that is used in default of a specified value; that is, the value the system assumes if the parameter is not coded.

Use the parameters to specify the services and options to be performed, and write them according to the following rules:

- If the selected parameter is written in all capital letters (for example, PROFILE or ENTITY or ENTITYX), code the parameter exactly as shown.
- If the selected parameter is written in italics, substitute the indicated value, address, or name.
- If the selected parameter is a combination of capital letters and italics separated by an equal sign (for example, VOLSER=*vol addr*), code the capital letters and equal sign as shown, and then make the indicated substitution for the italics.
- Code commas and parentheses exactly as shown.
- Positional parameters (parameters without equal signs) appear first; you must code them in the order shown. You may code keyword parameters (parameters with equal signs) in any order.
- If you select a parameter, read the third column before proceeding to the next parameter. The third column often contains coding restrictions for the parameter.

Continuation Lines

You can continue the parameter field of a macro instruction on one or more additional lines according to the following rules:

1. Enter a continuation character (not blank, and not part of the parameter coding) in column 72 of the line.
2. Continue the parameter field on the next line, starting in column 16. All columns to the left of column 16 must be blank.

You can code the parameter field being continued in one of two ways. Either code the parameter field through column 71, with no blanks, and continue in column 16 of the next line, or truncate the parameter field with a comma, where a comma normally falls, with at least one blank before column 71, and then continue in column 16 of the next line. Figure 2 on page 5 shows an example of each method.

```
1         10    16                          44                          72
|         |     |                           |                           |
↓         ↓     ↓                           ↓                           ↓
NAME1     OP1   OPERAND1,OPERAND2,OPERAND3,OPERAND4,OPERAND5,OPERAND6,OPX
                ERAND7                      THIS IS ONE WAY
NAME2     OP2   OPERAND1,OPERAND2,          THIS IS ANOTHER WAY         X
                OPERAND3,OPERAND4,                                      X
                OPERAND5,OPERAND6,OPERAND7
```

Figure 2. Continuation Coding

Chapter 2. RACF System Macros

This chapter contains the external RACF system macros that other callers can use to invoke RACF or another security product.

The RACF system macros are received as part of the z/VM program product; installations receive these macros even if they do not intend to install RACF. The RACROUTE macro instruction is the interface for all products that provide resource control. An external security product must be installed and active in order to use any of the RACROUTE interface function.

The following lists the RACF macros that you can invoke with the full function RACROUTE interface. IBM recommends that installations use the full function RACROUTE interface instead of the independent RACF system macros. Keywords and macro invocations introduced after Release 1.8.2 are supported only if you invoke them using this RACROUTE interface.

- "RACROUTE REQUEST=AUDIT: General-Purpose Security-Audit Request" on page 27 is used to audit requests to use a function or access a resource without authorization checking.
- "RACROUTE REQUEST=AUTH: Check RACF Authorization" on page 35 is used to provide authorization checking when a user requests to use a function or access a resource.
- "RACROUTE REQUEST=DEFINE: Define, Modify, Rename, or Delete a Resource for RACF" on page 55 is used to define, modify, or delete resource profiles for RACF.
- "RACROUTE REQUEST=DIRAUTH: Check RACF-Directed Authorization to a Sent Message" on page 86 is used to perform security label authorization checking on messages for installations using SECLABELs.
- "RACROUTE REQUEST=EXTRACT: Replace or Retrieve Fields" on page 92 is used to retrieve or update specified resource profile fields or to encode data.
- "RACROUTE REQUEST=FASTAUTH: Verify Access to Resources" on page 121 is used to provide authorization checking when a user requests access to a RACF-protected resource similar to RACROUTE REQUEST=AUTH. However, RACROUTE REQUEST=FASTAUTH verifies access to resources that have RACF profiles brought into main storage by the REQUEST=LIST macro service. RACF does not perform any auditing with this request.
- "RACROUTE REQUEST=LIST: Build In-Storage Profiles" on page 128 is used to retrieve general resource profiles and build an in-storage list for faster authorization checking. The list is attached to the ACEE.
- "RACROUTE REQUEST=STAT: Determine RACF Status" on page 138 is used to determine if RACF or another security product is active and, optionally, to determine whether protection is in effect for a given resource class. REQUEST=STAT can also be used to determine if a resource class name is defined.
- "RACROUTE REQUEST=TOKENBLD: Build a UTOKEN" on page 144 is used to modify an existing token.
- "RACROUTE REQUEST=TOKENMAP: Access Token Fields" on page 153 is used to convert a user token (UTOKEN) or a resource token (RTOKEN) into either internal or external format.
- "RACROUTE REQUEST=TOKENXTR: Extract UTOKENS" on page 158 is used to extract a UTOKEN from the current task or address space ACEE.
- "RACROUTE REQUEST=VERIFY: Identify and Verify a RACF-Defined User" on page 162 is used to provide user identification and verification.
- "RACROUTE REQUEST=VERIFYX: Verify User and Return a UTOKEN" on page 185 is used to create a user token (UTOKEN) for a unit of work. It provides for propagation of USERID, GROUPID, and SECLABEL for locally submitted jobs and is similar to VERIFY in some respects.

The following lists RACF system macros which are invoked independently of the RACROUTE interface.

- "RACSYNC Macro (z/VM Only)" on page 204 is used on z/VM to access the returned RACROUTE parameter list and load registers as if the call were synchronous.

© Copyright IBM Corp. 1990, 2022

RACROUTE: Router Interface

The RACROUTE macro is the interface to RACF (or another external security manager) for z/VM resource managers. The macro descriptions in this book define this interface. This does not imply that the z/VM operating system supports all the functions allowed by the interface. Rather, it defines the macros and keywords that are available for z/VM resource managers to implement security for data and other resources.

- **On z/VM:**

  A service machine that is running a CMS-based application, for example, can request a user's authority to a specific resource. The application service machine invokes RACROUTE to send the request to the RACF service machine. The RACF service machine makes a decision on the request and sends the response to the application service machine that made the request.

  Keywords that are used specifically to support the RACROUTE implementation on z/VM are identified as such and require that RELEASE=1.9 or a later release number be specified.

  When you use RACROUTE on CMS, communication between the RACF service machine and the invoker of the RACROUTE request uses CMSIUCV macro invocations. Installations planning to use RACROUTE must ensure their applications can coexist with CMSIUCV.

  When you use RACROUTE on GCS, communication between the RACF service machine and the invoker of the RACROUTE request uses IUCVINI and IUCVCOM macro invocations. Installations planning to use RACROUTE must ensure their applications can coexist with IUCVINI and IUCVCOM macro invocations.

  For more information on using RACROUTE on z/VM, see “Special Considerations for Using RACROUTE on z/VM” on page 12.

In coding the RACROUTE macro to perform a particular request, you must also use the necessary parameters for that request type on the RACROUTE macro instruction. For example, if you code RACROUTE to access REQUEST=AUTH, you must code REQUEST=AUTH and any other required parameters as well as any optional ones you need from the RACROUTE REQUEST=AUTH macro. RACROUTE ensures that only the parameters applicable to the RACROUTE REQUEST=AUTH request type have been coded.

Note: With Release 1.9 or later, when the function verifies the parameter list, if a keyword other than SEGDATA, STOKEN, TOKNIN, or TOKNOUT has an associated length and this length is zero, the function assumes that the keyword is not specified. For the SEGDATA keyword on the RACROUTE REQUEST=EXTRACT, TYPE=REPLACE, a length of zero is valid. For STOKEN, TOKNIN, and TOKNOUT, the keyword is considered not specified if both the associated length and version fields are zero in the token area.

Table 1 on page 8 identifies the system macro request types that are replacements for the independent RACF system macros documented in Appendix A, “Independent RACF System Macros,” on page 207.

Table 1. RACROUTE REQUEST=type and Independent RACF System Macros

RACROUTE Request Type	Equivalent Independent RACF System Macro
REQUEST=AUTH	RACHECK
REQUEST=DEFINE	RACDEF
REQUEST=EXTRACT	RACXTRT
REQUEST=FASTAUTH	FRACHECK
REQUEST=LIST	RACLIST
REQUEST=STAT	RACSTAT
REQUEST=VERIFY	RACINIT

Keyword and Parameter Cross-Reference for RACROUTE

Table 2 on page 9 lists the parameters available through the RACROUTE macro interface and cross-references them to each REQUEST=*type* in the entire RACROUTE macro. The allowable parameters for each REQUEST=*type* are marked with an "X". See the specific RACROUTE REQUEST=*type* macro descriptions for information about the use of the parameters for each type of request.

Table 2. RACROUTE REQUEST=keyword Cross-Reference

RACROUTE Parameters	AUDIT	AUTH	DEFINE	DIRAUTH	EXTRACT	FASTAUTH	LIST	STAT	TOKENBLD	TOKENMAP	TOKENXTR	VERIFY	VERIFYX
ACCLVL=		X	X										
ACEE=	X	X	X		X	X	X	X			X	X	
ACTINFO=												X	X
APPL=		X				X	X	X				X	X
ATTR=		X				X							
AUDIT=			X										
CHKAUTH=			X										
CLASS=	X	X	X		X	X	X	X					
DATA=			X										
DATEFMT=					X								
DECOUPL=	X	X	X	X	X	X	X	X	X	X	X	X	X
DERIVE=					X								
DSTYPE=		X	X										
ECB1=	X	X	X	X	X	X	X	X	X	X	X	X	X
ECB2=	X	X	X	X	X	X	X	X	X	X	X	X	X
ENCRYPT=					X							X	X
ENTITY=		X	X		X	X							
ENTITYX=	X	X	X		X								
ENTRY=								X					
ENVIR=			X				X					X	
ENVRIN=												X	
ENVROUT=												X	
ERASE=			X										
EVENT=	X												
EVQUAL=	X												
EXENODE=									X			X	X
EXPDT=			X										
EXPDTX=			X										
FIELDS=					X								
FILESEQ=		X	X										
FILTER=							X						
FLDACC=					X								
FORMOUT=										X			
GENERIC=		X	X		X								

Table 2. RACROUTE REQUEST=keyword Cross-Reference (continued)

RACROUTE Parameters	AUDIT	AUTH	DEFINE	DIRAUTH	EXTRACT	FASTAUTH	LIST	STAT	TOKENBLD	TOKENMAP	TOKENXTR	VERIFY	VERIFYX
GROUP=								X	X			X	X
GROUPID=		X											
INSTLN=		X	X			X	X					X	X
JOBNAME=												X	X
LEVEL=			X										
LIST=							X						
LOG=		X		X								X	X
LOGSTR=	X	X										X	X
MATCHGN=					X								
MCLASS=			X										
MENTITY=			X										
MENTX=			X										
MF=	X	X	X	X	X	X	X	X	X	X	X	X	X
MGENER=			X										
MGMTCLA=			X										
MSGRTRN=	X	X	X	X	X	X	X	X	X	X	X	X	X
MSGSP=	X	X	X	X	X	X	X	X	X	X	X	X	X
MSGSUPP=	X	X	X	X	X	X	X	X	X	X	X	X	X
MVOLSER=			X										
NEWNAME=			X										
NEWNAMX=			X										
NEWPASS=												X	X
NOTIFY=			X										
OLDVOL=		X											
OWNER=			X				X						
PASSCHK=												X	X
PASSWRD=												X	X
PGMNAME=												X	X
POE=									X			X	X
POSTEXI=	X	X	X	X	X	X	X	X	X	X	X	X	X
PREEXI=	X	X	X	X	X	X	X	X	X	X	X	X	X
RACFIND=		X	X										
RECVR=		X											
RELATED=	X	X	X	X	X	X	X	X	X	X	X	X	X
RELEASE=	X	X	X	X	X	X	X	X	X	X	X	X	X
REMOTE=									X			X	X
REQSTOR=	X	X	X	X	X	X	X	X	X	X	X	X	X
RESOWN=			X										
RESULT=	X												

Table 2. RACROUTE REQUEST=keyword Cross-Reference (continued)

RACROUTE Parameters	AUDIT	AUTH	DEFINE	DIRAUTH	EXTRACT	FASTAUTH	LIST	STAT	TOKENBLD	TOKENMAP	TOKENXTR	VERIFY	VERIFYX
RETPD=			X										
RTOKEN=		X		X									
SECLABL=			X						X			X	X
SECLVL=			X										
SEGDATA=					X								
SEGMENT=					X								
SESSION=									X			X	X
SGROUP=									X			X	X
SNODE=									X			X	X
STAT=												X	X
STATUS=		X											
STOKEN=									X			X	X
STORCLA=			X										
SUBPOOL=					X							X	
SUBSYS=	X	X	X	X	X	X	X	X	X	X	X	X	X
SUSERID=									X			X	
TAPELBL=		X	X										
TERMID=									X			X	X
TOKNIN=									X	X		X	X
TOKNOUT=									X	X	X	X	X
TRUSTED=									X			X	X
TYPE=			X		X								
UACC=			X										
UNIT=			X										
USERID=		X							X			X	X
USERWRD=	X	X	X	X	X	X	X	X	X	X	X	X	
UTOKEN=		X											
VOLSER=		X	X		X								
WARNING=			X										
WKAREA=						X							
WORKA=	X	X	X	X	X	X	X	X	X	X	X	X	X

Addressing Considerations

If a caller is executing in 24-bit addressing mode, all parameters and parameter lists are assumed to reside below 16MB. If a caller, however, is executing in 31-bit addressing mode, all parameters and parameter lists may reside above 16MB (that is, all parameter addresses are 31-bit addresses).

All parameter lists generated by the RACROUTE macro are in a format that allows assembled code to be moved above 16MB without being reassembled.

Special Considerations for Using RACROUTE on z/VM

The following section describe information you must consider before using RACROUTE on your z/VM system.

Authorization to Issue RACROUTE Requests

Use the following procedure to authorize virtual machines to issue RACROUTE requests. This authorization applies to all RACROUTE requests that specify RELEASE=1.9 or any later release. (This authorization does not apply to RACROUTE requests issued within the RACF service machine.)

You should limit the number of virtual machines that are authorized to use the RACROUTE interface on z/VM. The performance of RACF may be affected if many virtual machines are issuing RACROUTE requests to the RACF service machine.

1. Identify the RACF service machine to which RACROUTE requests will be sent.
   - Make sure the issuer of the RACROUTE requests has access to the RACF SERVMACH file. This file is placed on the CMS Y-disk during RACF installation, and the default is to send RACROUTE requests to RACFVM.
   - If you want the RACROUTE requests to be sent to a RACF service machine other than RACFVM, copy the RACF SERVMACH file to another minidisk accessed by the RACROUTE issuer. Change the user ID in the RACF SERVMACH file to the user ID of the RACF service machine to which you want the RACROUTE requests sent.
   - You may want to dedicate a RACF service machine to process RACROUTE requests.

     See *z/VM: RACF Security Server System Programmer's Guide* for more information.
2. Make sure the RACROUTE issuer has IUCV authorization by performing one of these two steps:
   a. To provide global IUCV authorization, so any user in the system can connect to the RACF service machine, update the RACF service machine's CP directory entry by adding this IUCV statement:

      ```
      IUCV ALLOW
      ```

   b. To give IUCV authorization to a single user, update the RACROUTE issuer's CP directory entry by adding an IUCV statement that specifies the RACF service machine with which the RACROUTE issuer will be communicating, for example:

      ```
      IUCV RACFVM PRIORITY MSGLIMIT 255
      ```

   See *z/VM: Planning and Administration* for more information.
3. RACF-authorize a connection to the RACF service machine
   - Log on with a user ID having the system-SPECIAL attribute
   - Create a profile named ICHCONN in the FACILITY class:

     ```
     RDEFINE FACILITY ICHCONN UACC(NONE)
     ```

   - Give READ or UPDATE access authority to appropriate service machines:

     ```
     PERMIT ICHCONN CLASS(FACILITY) ID(user-ID|group-ID)
            ACCESS(appropriate-access)
     ```

     where *appropriate-access* is one of the following:

     NONE
     Prevents use of the RACROUTE macro

     READ
     Allows use of the RACROUTE macro request types that are less sensitive in nature. That is, any RACROUTE request which could execute without APF authority on a z/OS system (such as REQUEST=AUTH).

UPDATE
Allows use of the RACROUTE macro request types that are more sensitive in nature. That is, any RACROUTE request which could not execute without APF authority on a z/OS system (such as REQUEST=VERIFY).

Note: Refer to the description for the specific request to determine which level of authority is required.

- Activate the FACILITY class (if this class is not already active):

```
SETROPTS CLASSACT(FACILITY)
```

4. Follow the procedures described in "Issuing RACROUTE Requests on CMS" on page 13 or "Issuing RACROUTE Requests on GCS" on page 13 to set up the environment to issue RACROUTE requests on CMS or GCS, respectively.

   The RPIUCMS and RPIUGCS modules referred to in the procedures are available in the RACF product, not the z/VM operating system. If you install another external security product on z/VM, that external security product should provide equivalent RPIUCMS and RPIUGCS functions as described in Appendix C, "RACROUTE Interface to an External Security Manager Product (Non-RACF) on z/VM," on page 337.

Issuing RACROUTE Requests on CMS

Before issuing RACROUTE requests on CMS, the user machine must set up an environment by executing the following RACF module:

```
RPIUCMS INIT
```

The following message is issued if the module execution is successful:

```
RPICMS016I USER/RACF z/VM communication path established
```

The user machine can now issue RACROUTE requests. After the RACROUTE requests are issued, the user machine should execute the following RACF module:

```
RPIUCMS TERM
```

When the user machine is logged off, the communication path is terminated. When the user machine is logged back on, the communication path must be re-established with RPIUCMS INIT.

Note: The RPIUCMS module runs as a nucleus extension and is placed on the CMS file mode Y when RACF is installed on z/VM.

Issuing RACROUTE Requests on GCS

To issue RACROUTE requests on GCS, the user must have access to supervisor state and authorized GCS functions. For more information see the *z/VM: Group Control System* book's "Planning" section.

Before issuing RACROUTE requests on GCS, the user machine must set up an environment for the requests to be processed. Each authorized member of the GCS group who will be issuing RACROUTE requests must do the following:

1. Issue GCS command:

```
GLOBAL LOADLIB RPIGCS
```

2. Issue GCS command:

```
LOADCMD RPIUGCS RPIATGCS
```

3. Issue GCS command:

```
LOADCMD RPISSSRC RPISSSRC
```

4. Execute the RACF module:

```
RPISSSRC
```

5. Execute the RACF module:

```
RPIUGCS INIT
```

The following message is issued if the command is successful:

```
RPIGCS012I USER/RACF z/VM communication path established
```

The user machine can now issue RACROUTE requests. After the RACROUTE requests are issued, the user machine should execute the following RACF module:

```
RPIUGCS TERM
```

When the user machine is logged off, the communication path is terminated. When the user machine is logged back on, the communication path must be re-established using the procedure described above.

Event Control Blocks (ECBs) and Their Significance on z/VM

- No event-control-block (ECB) address specified

  If the application program does not specify an ECB address on the RACROUTE macro invocation, when control is returned to the application program, registers 15, 0, and 1 are set with the answer to the RACROUTE request. The scenario is analogous to issuing a RACROUTE request on z/OS.
- One ECB address specified

  Upon return of the RACROUTE invocation, register 15 contains a return code. This return code is not the answer to the request; rather, it indicates if the request has been accepted for processing by RACF. Return code 0 means that the request was successfully sent to RACF. Any return code other than 0 means that RACF was not available to process the request.

  If the request has been accepted, the application can issue a WAIT macro instruction, which waits for the ECB.

  When RACF processes the request and sends the response, the ECB is posted using the POST macro to signal completion of the request. This notifies the application program that the answer to the RACROUTE request is available.
- Two ECB addresses specified

  Upon return of the RACROUTE invocation, register 15 contains a return code. This return code is not the answer to the request, but rather an indication of whether the RACROUTE request has been accepted for processing by RACF. Return code 0 means that the request was successfully sent to RACF. Any return code other than 0 means that RACF was not available to process the request.

  If the request has been accepted, the application can go on to other tasks while the particular task for which the RACROUTE was invoked goes into a wait state.

  When RACF processes the request and sends the response, both ECBs are posted with the mask X'40008000'. (The POST macro is not used.) This notifies the application program that the answer to the RACROUTE request is available.

  As the application program is processing other tasks, it periodically checks the high-order bits of the two ECBs to see whether RACF has returned a decision. The other tasks that are being processed in the machine do not have to wait until that particular RACROUTE request is processed.

When either one or two ECBs are specified, the answer to the RACROUTE invocation is contained in the beginning of the RACROUTE-request parameter list. The application program must use either the RACSYNC macro or the ICHSAFP macro to set up addressability to the RACROUTE parameter list to examine the following fields and so determine the disposition of the RACROUTE request:

- RACF return and reason codes

  - SAFPRRET
  - SAFPRREA
- RACROUTE return and reason codes
  - SAFPSFRC
  - SAFPSFRS

  The RACROUTE return and reason codes from a z/VM invocation of RACROUTE are treated the same as the SAF return and reason codes from an z/OS invocation of RACROUTE.
- RACF returned data
  - SAFPRETD

  This field contains the address of returned data from RACROUTE.

ACIGROUP Considerations for the ENTITY Parameter

For creation of entity names pertaining to the classes VMRDR or VMMDISK, the following condition applies in the z/VM environment.

If the resource class for an authorization check is VMMDISK or VMRDR, the issuer of RACROUTE must issue a DIAGNOSE code X'A0', subcode X'00', to extract the ACIGROUP of the requesting user.

If an ACIGROUP exists, the entity name must be prefixed with the ACIGROUP before issuing the RACROUTE request.

For information on using DIAGNOSE code X'A0', subcode X'00', refer to *z/VM: CP Programming Services*.

Example 1

A resource manager wants to determine whether USERA has access to USERB's 191 minidisk. USERB is connected to ACIGROUP GROUP1. The invoker builds the RACROUTE entity name as follows:

```
USERB.191
```

DIAGNOSE code X'A0', subcode X'00', must now be issued to determine whether USERB is associated with an ACIGROUP. If the diagnose return code indicates no connection, the entity name must remain as USERB.191.

If the diagnose indicates an ACIGROUP connection, the entity name must be prefixed with the data returned by the diagnose instruction, as in

```
GROUP1.USERB.191
```

Example 2

A resource manager wants to determine whether USERA has access to USERB's virtual reader. USERB is connected to ACIGROUP GROUP1. The invoker builds the RACROUTE entity name as follows:

```
USERB
```

DIAGNOSE code X'A0', subcode X'00', must now be issued to determine whether USERB is associated with an ACIGROUP. If the diagnose return code indicates no connection, the entity name must remain as USERB.

If the diagnose indicates an ACIGROUP connection, the entity name must be prefixed with the data returned by the diagnose instruction.

```
GROUP1.USERB
```

RACROUTE (Standard Form)

The standard form of the RACROUTE macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=*type*	*type*: System macro request type
,WORKA=*work area addr*	*work area addr*: A-type address or register (2) - (12)
,DECOUPL=YES ,DECOUPL=NO	**Default:** DECOUPL=NO
,MSGRTRN=YES ,MSGRTRN=NO	**Default:** MSGRTRN=NO
,MSGSP=*subpool number*	*subpool number*: Decimal digit 0-255; default is 0.
,MSGSUPP=YES ,MSGSUPP=NO	**Default:** MSGSUPP=NO
,RELATED=*value*	*value*: Any valid macro keyword specified
,REQSTOR=*reqstor addr*	*reqstor addr*: A-type address or register (2) - (12) **Note:** If you specify REQSTOR and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.
,SUBSYS=*subsys addr*	*subsys addr*: A-type address or register (2) - (12) **Note:** If you specify SUBSYS and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.

THE FOLLOWING KEYWORDS APPLY ONLY IN THE CMS and GCS ENVIRONMENTS

,ECB1=*ECB1 address*	*ECB1 address*: A-type address or register (2) - (12)
,ECB2=*ECB2 address*	*ECB2 address*: A-type address or register (2) - (12)
,POSTEXI=*postprocessing exit address*	*postprocessing exit address*: A-type address or register (2) - (12)
,PREEXI=*preprocessing exit address*	*preprocessing exit address*: A-type address or register (2) - (12)
,USERWRD=*request identifier address*	*request identifier address*: A-type address or register (2) - (12)

For RACROUTE to work correctly, once you have chosen a REQUEST, you must also specify the parameters that belong to that request. Please see the RACROUTE REQUEST= macros for the necessary parameters.

This request requires a standard 18-word save area that is pointed to by register 13.

Data areas returned to the caller by RACF are either above or below 16MB, depending upon the caller's addressing mode and the data area in question.

The parameters are explained as follows:

,REQUEST=AUDIT
,REQUEST=AUTH
,REQUEST=DEFINE
,REQUEST=DIRAUTH
,REQUEST=EXTRACT
,REQUEST=FASTAUTH
,REQUEST=LIST
,REQUEST=STAT
,REQUEST=TOKENBLD
,REQUEST=TOKENMAP
,REQUEST=TOKENXTR
,REQUEST=VERIFY
,REQUEST=VERIFYX
specifies the system macro request type.

To invoke a system macro request supported through the RACROUTE interface, you must also code the parameters associated with that particular request type on the RACROUTE macro instruction. See the extended description for the system macro request type you want to use for specific information about the keywords available for that particular request.

,WORKA=*work area addr*
specifies the address of a 512-byte work area for use by the router and the RACF front-end routine. This parameter is required for execution of the RACROUTE macro. Where it is specified can vary. For example, on MF=S it must be specified on the MF=S invocation. For MF=E, it may be specified on the MF=E invocation, on an earlier MF=M invocation that points to the same parameter list, or on the MF=L invocation that built the parameter list.

,DECOUPL=YES
,DECOUPL=NO
specifies whether or not REQSTOR and SUBSYS are to be used for caller identification or to determine whether RACF function is to be performed or bypassed. (See the SUBSYS keyword.)

DECOUPL=YES specifies that REQSTOR and SUBSYS parameters do not require corresponding entries in the router table because they are to be used only for caller identification.

DECOUPL=NO specifies that REQSTOR and SUBSYS parameters must have corresponding entries in the router table because they are to be used not only for identification, but also to determine whether RACF function is to be performed or bypassed.

To use this keyword, you must specify RELEASE=1.9 or a later release number.

,ECB1=*ECB1 address*
specifies the address of the first event control block (ECB) to be processed. This keyword is used to notify the RACROUTE invoker that the request has been processed.

This keyword applies to z/VM only.

,ECB2=*ECB2 address*
specifies the address of the second event control block (ECB). This keyword is used to notify the RACROUTE invoker that the request has been processed.

This keyword applies to z/VM only.

,MSGRTRN=YES
,MSGRTRN=NO
specifies whether you want to use message return processing. You can use this parameter in conjunction with the other MSG*xxxx* parameters to control the disposition of messages generated by this service.

To use this parameter, you must also specify RELEASE=1.8 or a later release number.

Note:

1. This parameter applies to REQUEST=AUTH, REQUEST=DEFINE, REQUEST=VERIFY, and REQUEST=VERIFYX.
2. IRR102I, IRR101I, ICH408I, ICH70001I, ICH70002I, ICH70003I, ICH70004I, ICH70005I, ICH70006I, and ICH70007I messages can be returned for RACF.
3. When control returns from RACROUTE, the RACROUTE parameter-list field is mapped by SAFPMSAD in the ICHSAFP mapping macro. SAFPMSAD is nonzero if messages have been returned. This field will contain the address of an area that consists of two fullwords followed by the message itself in write-to-operator (WTO) parameter-list format. The first word is the length of the area including the two-fullword header; the second word points to the next message area, if there is one, or contains zero if no more messages areas exist.

   You must issue the FREEMAIN macro to release the message area.

,MSGSP=*subpool number*
specifies the storage subpool into which you want RACF messages returned. You can use this parameter in conjunction with the other MSG*xxxx* parameters to control the disposition of messages generated by this service. If you do not specify a subpool, the default subpool is 0.

On z/VM in the CMS Environment: If you specify a subpool, you must adhere to the subpools supported by the CMS/OS simulation of GETMAIN. For more information, see the *z/VM: CMS Application Development Guide for Assembler*.

On z/VM in the GCS Environment: If you specify a subpool, you must adhere to the subpools supported by GCS. For more information, see the *z/VM: Group Control System*.

Note: This parameter applies to REQUEST=AUTH, REQUEST=DEFINE, REQUEST=VERIFY, and REQUEST=VERIFYX.

,MSGSUPP=YES
,MSGSUPP=NO
specifies whether you want to suppress WTO messages from within RACF processing. You can use this parameter in conjunction with the other MSG*xxxx* parameters to control the disposition of messages generated by this service.

To use this parameter, you must also specify RELEASE=1.8 or a later release number.

Note: This parameter applies to REQUEST=AUTH, REQUEST=DEFINE, REQUEST=VERIFY, and REQUEST=VERIFYX, and, for RACF only, message ICH408I and associated auditing support informational messages (IRR series), as well as ICH70001I, ICH70002I, ICH70003I, ICH70006I, and ICH70007I.

,POSTEXI=*postprocessing exit address*
specifies the address of a postprocessing exit routine that is given control after the RACF service machine responds to the RACROUTE request but before the user machine regains control. If the request is not sent to the RACF service machine (usually because of an error condition), the postprocessing exit is still given control before the user machine regains control.

On an asynchronous RACROUTE request (ECB1= or ECB2= is specified), if the return code from the exit is anything other than zero, RACF does not post the ECB because it is assumed the application has done so.

The exit runs in the user's virtual machine, and the application must load the exit. The exit receives control through a BALR instruction, with register 1 pointing to a fullword that contains the address of the RACROUTE parameter list (mapped by ICHSAFP).

This keyword applies to z/VM only.

,PREEXI=*preprocessing exit address*
specifies the address of a preprocessing exit routine that is given control in the invoker's machine before sending the request to the RACF service machine for processing.

On an asynchronous RACROUTE request (ECB1= or ECB2= is specified), if the return code from the exit is anything other than zero, RACF does not post the ECB because it is assumed the application has done so.

The exit runs in the user's virtual machine, and the application must load the exit. The exit receives control through a BALR instruction, with register 1 pointing to a fullword that contains the address of the RACROUTE parameter list (mapped by ICHSAFP).

This keyword applies to z/VM only.

,RELATED=*value*
specifies information used to make notes to yourself to document macro instructions by relating functions or services to corresponding functions or services. You can use any format and put in any length and type of data you want.

,REQSTOR=*reqstor addr*
specifies the address of an 8-byte character field containing the name of the piece of code that is making the request. (This address identifies a unique piece of code within a set of code that exists in a subsystem.) If this operand is omitted, a string of eight blanks is assumed.

Beginning with Release 1.9, you do not have to put a matching entry in the router table if you specify the DECOUPL keyword. (See the DECOUPL keyword.)

Before Release 1.9, if you specified this operand and RACF was installed, you had to update the RACF router table with a matching entry. If you did not update the table, RACF processing was bypassed. For a description of the RACF router table and the macro used to update it, see "ICHRFRTB Macro" in *z/VM: RACF Security Server Macros and Interfaces*.

,SUBSYS=*subsys addr*
specifies the address of an 8-byte character field containing the calling subsystem's name, version, and release level. If this operand is omitted, a string of eight blanks is assumed.

Beginning with Release 1.9, you do not have to put a matching entry in the router table if you specify the DECOUPL keyword. (See the DECOUPL keyword.)

Before Release 1.9, if you specified this operand and RACF was installed, you had to update the RACF router table with a matching entry. If you did not update the table, RACF processing was bypassed.

For a description of the RACF router table and the macro used to update it, see "ICHRFRTB Macro" in *z/VM: RACF Security Server Macros and Interfaces*.

,USERWRD=*request identifier address*
specifies 4 bytes of data you can use to identify the specific RACROUTE request being made. The data can be of any format and type.

Specific to the GCS environment, GCS provides a multitasking capability that allows a GCS-based resource manager to generate more than one task in the same virtual machine. It may be important to the resource manager to be able to create a separate ACEE to represent the same user in more than one of these separate tasks. Ordinarily, the ACEE created by RACROUTE REQUEST=VERIFY processing for one task overlays the ACEE created by RACROUTE REQUEST=VERIFY processing for the task before it, if the request is being issued from the same virtual machine. For example, TASK1 performs a RACROUTE REQUEST=VERIFY and creates an ACEE; then TASK2 performs a RACROUTE REQUEST=VERIFY for the same user ID and creates an ACEE; this ACEE overlays the ACEE created by TASK1. Should TASK1 perform a RACROUTE REQUEST=AUTH, that RACROUTE REQUEST=AUTH uses the ACEE created by TASK2, which would be incorrect.

To prevent this from happening, whenever you need to maintain more than one ACEE concurrently for a user in the same GCS machine, you must specify a USERWRD on the RACROUTE REQUEST=VERIFY to create the ACEE. You must do this for all subsequent RACROUTE invocations for that user. USERWRD is the key that RACF uses to distinguish and manage the ACEEs that represent a single user.

In the CMS environment, RACF ignores this keyword.

This keyword applies to z/VM only.

Return Codes

These return codes represent return codes from all invocations of the RACROUTE macro; for example, REQUEST=AUTH, REQUEST=VERIFY. For specific information on the success or failure of the invocation in question, see the section of this book that describes that invocation.

When you execute the macro, space for RACF return codes and their respective reason codes is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains one of the following SAF return codes.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
The requested security function completed successfully. For example, if the requested function was ‘AUTH’, the authorization request was accepted.

04
The requested function has not been processed. For example, if the request was ‘AUTH’, RACF or the SAF router could neither accept nor fail the request. The following are some possible reasons for a request's not being processed.

- The SAF router is not active.

- The RACF front-end routine detected that a null action was requested for the specified request type, resource type, and subsystem ID.
- The combination of request, resource, and subsystem could not be found in the RACF router table.
- RACF is not active on the system, and RACFIND=YES was not specified, and there is no RACROUTE installation exit routine (or an exit originated a return code of 4).
- RACF is active on the system, but no profile exists for the specified resource.
- The class is not defined to RACF.

08
The requested function was processed by RACF, the SAF router, or the router exit (ICHRTX00) and failed. (**On z/VM,** ICHRTX00 is a RACF exit.) If the requested function was AUTH, the authorization request has failed. For example, if RACF is inactive for an 'AUTH' request with RACFIND=YES, the SAF router fails the request. The RACF- or router-exit return code and reason codes are returned in the first two words of the RACROUTE input parameter list.

Note: On z/VM, RACF performs the function of the SAF router.

Additional Return Codes and Reason Codes (z/VM only)

The following return and reason codes are issued only in the z/VM environment. They can be issued for any RACROUTE request.

Note: Some RACROUTE parameter errors result in a RACF abend. **On z/VM,** these abends are simulated by RACF return and reason codes. The RACF return and reason codes reflect the abend reason codes documented in *z/VM: RACF Security Server Messages and Codes*.

Note:
All return and reason codes are shown in hexadecimal.

RACF Return Code
Meaning

FFF
Processing could not be completed in the RACF service machine.

RACF Reason Code
Meaning

00
GETMAIN failed.

04
FREEMAIN failed.

08
IUCV RECEIVE failed.

0C
IUCV SEND failed.

10
The connection was severed.

14
The IUCV CONNECT failed.

18
A communication error occurred.

1C
An abend occurred in the RACF service machine.

20
The user is not authorized to issue this request.

FFE
Processing could not be completed in the user machine.

RACF Reason Code
Meaning

00
GETMAIN failed.

04
FREEMAIN failed.

08
IUCV RECEIVE failed.

0C
IUCV SEND failed.

10
The connection was severed.

14
The IUCV CONNECT failed.

18
A communication error occurred.

1C
An abend occurred in the RACF service machine.

20
The user is not authorized to issue this request.

Example 1

Operation: Invoke the SAF router to perform authorization checking, using the standard form, for a non-VSAM data set residing on the volume pointed to by register 8. Register 7 points to the data set name and the RACF user is requesting the highest level of control over the data set. The "RACF-indicated" bit in the data set's DSCB is on.

```
RACROUTE  REQUEST=AUTH,WORKA=RACWK,ENTITY=((R7)),        X
          VOLSER=(R8),CLASS='DATASET',ATTR=ALTER,        X
          RACFIND=YES
  .
  .
RACWK     DS  CL512
```

Example 2

Operation: Invoke the SAF router to perform authorization checking, using the standard form, for an IMS transaction pointed to by register 5. The user requests only read access.

```
RACROUTE  REQUEST=FASTAUTH,                              X
          WORKA=RACWK,ENTITY=(R5),                       X
          CLASS='TIMS',WKAREA=FRACWK,                    X
          ATTR=READ
  .
  .
  .
FRACWK    DS  16F
RACWK     DS  CL512
```

RACROUTE (List Form)

The list form of the RACROUTE macro is written as follows. Refer to the Standard Form of the RACROUTE macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute Form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=*type*	*type*: System macro request type
,WORKA=*work area addr*	*work area addr*: A-type address
,DECOUPL=YES ,DECOUPL=NO	**Default:** DECOUPL=NO
,MSGRTRN=YES ,MSGRTRN=NO	**Default:** MSGRTRN=NO
,MSGSP=*subpool number*	*subpool number*: Decimal digit 0-255; default is 0.
,MSGSUPP=YES ,MSGSUPP=NO	**Default:** MSGSUPP=NO
,RELATED=*value*	*value*: Any valid macro keyword specified
,REQSTOR=*reqstor addr*	*reqstor addr*: A-type address **Note:** If you specify REQSTOR and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.
,SUBSYS=*subsys addr*	*subsys addr*: A-type address **Note:** If you specify SUBSYS and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.

,MF=L

THE FOLLOWING KEYWORDS APPLY ONLY IN THE z/VM ENVIRONMENT

,ECB1=*ECB1 address*	*ECB1 address*: A-type address
,ECB2=*ECB2 address*	*ECB2 address*: A-type address
,POSTEXI=*postprocessing exit address*	*postprocessing exit address*: A-type address
,PREEXI=*preprocessing exit address*	*preprocessing exit address*: A-type address
,USERWRD=*request identifier address*	*request identifier address*: A-type address

The REQUEST= parameters are explained under their respective invocations. The RACROUTE parameters are explained under the standard form of the RACROUTE macro with the following exception:

,MF=L
specifies the list form of the RACROUTE macro instruction.

RACROUTE (Execute Form)

The execute form of the RACROUTE macro is written as follows. Refer to the Standard Form of the RACROUTE macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=*type*	*type*: System macro request type
,WORKA=*work area addr*	*work area addr*: Rx-type address or register (2) - (12)
,DECOUPL=YES	

,DECOUPL=NO

,MSGRTRN=YES
,MSGRTRN=NO

,MSGSP=*subpool number* — *subpool number*: Decimal digit 0-255

,MSGSUPP=YES
,MSGSUPP=NO

,RELATED=*value* — *value*: Any valid macro keyword specified

,REQSTOR=*reqstor addr* — *reqstor addr*: Rx-type address or register (2) - (12)

Note: If you specify REQSTOR and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.

,SUBSYS=*subsys addr* — *subsys addr*: Rx-type address or register (2) - (12)

Note: If you specify SUBSYS and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.

,MF=(E,*ctrl addr*) — *ctrl addr*: Rx-type address or register (1), (2) - (12)

THE FOLLOWING KEYWORDS APPLY ONLY IN THE z/VM ENVIRONMENT

,ECB1=*ECB1 address* — *ECB1 address*: Rx-type address or register (2) - (12)

,ECB2=*ECB2 address* — *ECB2 address*: Rx-type address or register (2) - (12)

,POSTEXI=*postprocessing exit address* — *postprocessing exit address*: Rx-type address or register (2) - (12)

,PREEXI=*preprocessing exit address* — *preprocessing exit address*: Rx-type address or register (2) - (12)

,USERWRD=*request identifier address* — *request identifier address*: Rx-type address or register (2) - (12)

The REQUEST= parameters are explained under their respective invocations. The RACROUTE parameters are explained under the standard form of the RACROUTE macro with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE macro where *ctrl addr* is the address of the associated parameter list.

RACROUTE (Modify Form)

The modify form of the RACROUTE macro is written as follows. Refer to the Standard Form of the RACROUTE macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=*type*	*type*: System macro request type
,WORKA=*work area addr*	*work area addr*: Rx-type address or register (2) - (12)
,DECOUPL=YES ,DECOUPL=NO	
,MSGRTRN=YES ,MSGRTRN=NO	
,MSGSP=*subpool number*	*subpool number*: Decimal digit 0-255
,MSGSUPP=YES ,MSGSUPP=NO	
,RELATED=*value*	*value*: Any valid macro keyword specified
,REQSTOR=*reqstor addr*	*reqstor addr*: Rx-type address or register (2) - (12) **Note:** If you specify REQSTOR and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.

,SUBSYS=*subsys addr*	*subsys addr*: Rx-type address or register (2) - (12) **Note:** If you specify SUBSYS and RACF is installed, you must either update the RACF router table to match the operand or specify DECOUPL=YES.
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1), (2) - (12)

THE FOLLOWING KEYWORDS APPLY ONLY IN THE z/VM ENVIRONMENT

,ECB1=*ECB1 address*	*ECB1 address*: Rx-type address or register (2) - (12)
,ECB2=*ECB2 address*	*ECB2 address*: Rx-type address or register (2) - (12)
,POSTEXI=*postprocessing exit address*	*postprocessing exit address*: Rx-type address or register (2) - (12)
,PREEXI=*preprocessing exit address*	*preprocessing exit address*: Rx-type address or register (2) - (12)
,USERWRD=*request identifier address*	*request identifier address*: Rx-type address or register (2) - (12)

The REQUEST= parameters are explained under their respective invocations. The RACROUTE parameters are explained under the standard form of the RACROUTE macro with the following exceptions:

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE macro, where *ctrl addr* is the address of the associated parameter list. The macro updates the parameter list, but does not execute the macro.

RACROUTE REQUEST=AUDIT: General-Purpose Security-Audit Request

The RACROUTE REQUEST=AUDIT macro is a general-purpose security-audit request that can be used to audit the specified resource name (ENTITYX) and action. This request records events in system-management-facilities (SMF) type 80 records, and issues messages to the network security administrator.

To use this service, you must also specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=AUDIT must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

SMF records are provided by RACF, not z/VM.

RACROUTE REQUEST=AUDIT (Standard Form)

The standard form of the RACROUTE REQUEST=AUDIT macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see "RACROUTE (Standard Form)" on page 16.

Note:
RACROUTE REQUEST=AUDIT requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=AUDIT. The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=AUDIT	
,EVENT=*'event name'* ,EVENT=*event name addr*	*event name*: 1- to 8-character name *event name addr*: A-type address or register (2) - (12)
,EVQUAL=*number* ,EVQUAL=*reg*	*number*: 0-99 *reg*: Register (2) - (12)
,RELEASE=*number*	*number*: 1.9.2, or 1.9 **Default:** RELEASE=1.6 **Note:** RACROUTE macro will not allow REQUEST=AUDIT to be specified unless RELEASE= is specified with a value of 1.9 or later.
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,CLASS=*'class name'* ,CLASS=*class name addr*	*class name*: 1- to 8-character name *class name addr*: A-type address or register (2) - (12)
,ENTITYX=*extended resource name addr*	*extended resource name addr*: A-type address or register (2) - (12)

,LOGSTR=*logstr addr*	*logstr addr*: A-type address or register (2) - (12)
,RESULT=SUCCESS	**Default:** RESULT=SUCCESS
,RESULT=FAILURE	
,MF=S	

The parameters are explained as follows:

,ACEE=*acee addr*
specifies the address of an ACEE passed on a REQUEST=AUDIT. RACF searches local profiles chained off the ACEE that have been placed there with the RACROUTE REQUEST=LIST macro.

The ACEE used should have been created as the result of a previous RACROUTE invocation (such as REQUEST=VERIFY,ENVIR=CREATE).

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies that you want RACF to perform authorization checking for a resource in this class. You can specify the class name or the class-name address. If you specify a class-name address, the address must point to an 8-byte field that contains the class name. The class name must be left-justified and padded to the right with blanks, if necessary.

For the event "GENERAL", REQUEST=AUDIT allows print service facility (PSF) on z/OS to perform auditing. Neither profiles nor settings specified in SETROPTS LOGOPTIONS are checked. It is assumed the requester always wants an SMF record cut. If the parameters are correct, auditing is always done.

For the class APPCLU, if SETROPTS LOGOPTIONS other than DEFAULT is specified, RACF uses the options to determine what auditing to perform. If SETROPTS LOGOPTIONS is set to DEFAULT, RACF searches the resource profile that matches the entity and uses the auditing options specified in the profile. If RACF does not find a corresponding profile, it does not perform any auditing. A message is issued to the network security administrator.

,ENTITYX=*extended resource name addr*
specifies the address of a structure that consists of two 2-byte length fields, followed by the entity name.

- The first 2-byte field specifies a buffer length that can be from 0 to 255 bytes. This length field refers to the length of the buffer that contains the entity name; it does not include the length of either length field.
- The second 2-byte field specifies the actual length of the entity name. This length field includes the length of the actual name without any trailing blanks; it does not include the length of either length field.

These two length fields can be used in several different ways:

- If you know the length of the entity name, you can specify 0 in the first field and the length of the entity name in the second field. When you specify the second field, note that each byte counts. This means the entity name that you specify must match exactly the entity name on the RACF database.
- If you choose to place the entity name in a buffer area, specify the length of the buffer in the first field. For the second field, do one of the following:
  - If you know the length of the entity name, specify the length in the second field. The length of the first field can be from 0 to 255, but *must* be equal to or greater than the length of the second field.
  - If you do not know the length of the entity name, specify 0 in the second field, and RACF will count the number of characters in the entity name.

,EVENT=*'event name'*
,EVENT=*event name addr*
specifies the name of the event that you want RACF to log. You can specify the event name or the event-name address. If you specify the event-name address, it must point to an 8-byte field that contains the event name. The event name must be left-justified and padded to the right with blanks.

The events that you can log with Release 1.9 or later are APPCLU (event code 26) and GENERAL (event code 27).

The event code GENERAL allows auditing of PSF security information. PSF uses qualify code 0 for this event. To achieve auditing, the LOGSTR and RESULT keywords should also be specified.

,EVQUAL=*number*
,EVQUAL=*reg*
specifies the event-code qualifier for the event that you want logged. If you specify a register rather than a number, you must enter the event-code qualifier in the low-order halfword of the register or the field the address in the register points to. With APPCLU, the qualifier can be from 0 to 12; with GENERAL, the qualifier can be from 0 to 99. See "SMF Records" in *z/VM: RACF Security Server Macros and Interfaces* for a description of RACF event-code qualifiers for an event.

,LOGSTR=*logstr addr*
specifies the address of a 1-byte length field followed by up to 255 bytes of character data that will be written to the SMF DATA file for z/VM, together with RACF audit information.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=AUDIT macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

RACROUTE macro will not allow REQUEST=AUDIT to be specified unless RELEASE= is specified with a value of 1.9 or later.

,RESULT=SUCCESS
,RESULT=FAILURE
specifies that the resource manager (for example, PSF) can specify a RESULT keyword that causes the audit record to be marked as a success or as a failure.

The default is RESULT=SUCCESS.

,MF=S
specifies the standard form of the RACROUTE REQUEST=AUDIT macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=AUDIT has completed successfully.

RACF Return Code
Meaning

00
The requested security function has completed successfully.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

04
The class is not active.

08
The requested function failed.

RACF Return Code
Meaning

08
The class was not specified or not defined to RACF.

0C
Indicates an internal error from RACXTRT.

Reason Code
Meaning

xxyy
xx is a return code from RACXTRT; *yy* is a reason code from RACXTRT.

10
Indicates parameter list-error as described by the following hex reason codes:

Reason Code
Meaning

00
Invalid event

04
Invalid event-code qualifier

08
Invalid parameter-list version

0C
Invalid parameter-list length

10
Invalid entity.

14
No auditing is done. One of the following is true:

- No profile is found and LOGOPTIONS is not set for this class.
- No profile is found and the class is included in a RACLIST.
- The class is not in a RACLIST.

Example

Operation: Invoke the RACROUTE REQUEST=AUDIT macro to search for a profile in the APPCLU class to match the entity specified in LULUPAIR. The profiles to be searched have been placed in storage using the RACROUTE REQUEST=LIST macro. Be aware that if SETROPTS LOGOPTIONS other than DEFAULT has been specified for the APPCLU class, those auditing options are the ones that RACF uses. Set the auditing options so that an SMF 80 event APPCLU event-code qualifier 04 (partner session keys were not equal) is logged. A message is sent to the security console, and message ICH70005I is sent to the caller.

Note: The message cannot be received by anyone other than the caller to which it was directed.

```
RACROUTE  REQUEST=AUDIT,CLASS='APPCLU',ENTITYX=LULUPAIR, X
          ACEE=VTAMACEE,EVENT='APPCLU',EVQUAL=CODE04,    X
          WORKA=RACWK,RELEASE=1.9
  .
  .
RACWK     DS  CL512
LULUPAIR  DS  0CL16
BUFLEN    DC  AL2(12)
ENTLEN    DC  AL2(12)
ENTITYX   DC  CL12'NET1.LU1.LU2'
```

RACROUTE REQUEST=AUDIT (List Form)

The list form of the RACROUTE REQUEST=AUDIT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=AUDIT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
ƀ	One or more blanks must precede RACROUTE.
RACROUTE	
ƀ	One or more blanks must follow RACROUTE.
REQUEST=AUDIT	
,RELEASE=*number*	*number*: See Standard Form and extended definition.
,ACEE=*acee addr*	*acee addr*: A-type address
,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address
,ENTITYX=*extended resource name addr*	*extended resource name addr*: A-type address
,EVENT=*'event name'*	*event name*: 1- to 8-character name
,EVENT=*event name addr*	*event name addr*: A-type address

,EVQUAL=*number*	*number*: 0-99
,LOGSTR=*logstr addr*	*logstr addr*: A-type address
,RESULT=SUCCESS ,RESULT=FAILURE ,MF=L	**Default:** RESULT=SUCCESS

The parameters are explained under the standard form of the RACROUTE REQUEST=AUDIT macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=AUDIT macro instruction.

RACROUTE REQUEST=AUDIT (Execute Form)

The execute form of the RACROUTE REQUEST=AUDIT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=AUDIT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
ƀ RACROUTE	One or more blanks must precede RACROUTE.
ƀ	One or more blanks must follow RACROUTE.
REQUEST=AUDIT	
,RELEASE=*number*	*number*: See Standard Form and extended description.
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ENTITYX=*extended resource name addr*	*extended resource name addr*: Rx-type address or register (2) - (12)
,EVENT=*event name addr*	*event name addr*: Rx-type address or register (2) - (12)

,EVQUAL=*number*	*number*: 0-99
,EVQUAL=*reg*	*reg*: Register (2) - (12)
,LOGSTR=*logstr addr*	*logstr addr*: Rx-type address or register (2) - (12)
,RESULT=SUCCESS	
,RESULT=FAILURE	
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=AUDIT macro with the following exception:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=AUDIT macro instruction.

RACROUTE REQUEST=AUDIT (Modify Form)

The modify form of the RACROUTE REQUEST=AUDIT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=AUDIT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
ƀ	One or more blanks must precede RACROUTE.
RACROUTE	
ƀ	One or more blanks must follow RACROUTE.
REQUEST=AUDIT	
,RELEASE=*number*	*number*: See Standard Form and extended description.
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ENTITYX=*extended resource name addr*	*extended resource name addr*: Rx-type address or register (2) - (12)
,EVENT=*event name addr*	*event name addr*: Rx-type address or register (2) - (12)

,EVQUAL=*number*	*number*: 0-99
,EVQUAL=*reg*	*reg*: Register (2) - (12)
,LOGSTR=*logstr addr*	*logstr addr*: Rx-type address or register (2) - (12)
,RESULT=SUCCESS	
,RESULT=FAILURE	
,MF=M	

The parameters are explained under the standard form of the RACROUTE REQUEST=AUDIT macro with the following exception:

,MF=M
specifies the modify form of the RACROUTE REQUEST=AUDIT macro instruction.

RACROUTE REQUEST=AUTH: Check RACF Authorization

The RACROUTE REQUEST=AUTH macro checks a user's authority to access a resource, based on a profile in the RACF database when a user requests access to a RACF-protected resource.

When RACF is installed, the caller of RACROUTE REQUEST=AUTH must have at least READ authority to access the ICHCONN profile in the FACILITY class. To specify the USERID and ACEE keywords, the caller must have at least UPDATE authority to access the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

RACROUTE REQUEST=AUTH (Standard Form)

The standard form of the RACROUTE REQUEST=AUTH macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

Note:
RACROUTE REQUEST=AUTH requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=AUTH. The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.

REQUEST=AUTH

,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,ENTITY=*resource name addr*	*resource name addr*: A-type address only
,ENTITY=(*resource name addr*)	*resource name addr*: A-type address or register (2) - (12)
,ENTITYX=*extended resource name addr*	*extended resource name addr*: A-type address only
,ENTITYX=(*extended resource name addr*)	*extended resource name addr*: A-type address or register (2) - (12)
,VOLSER=*vol addr*	*vol addr*: A-type address or register (2) - (12) **Note:** VOLSER is required for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used and when ENTITY is also coded.
,ACCLVL=(*access level addr*,*parm list addr*)	*parm list addr*: A-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address or register (2) - (12)
,ATTR=READ	**Default:** ATTR=READ
,ATTR=UPDATE	
,ATTR=CONTROL	
,ATTR=ALTER	
,ATTR=*reg*	*reg*: register (2) - (12)
,DSTYPE=N	**Default:** DSTYPE=N
,DSTYPE=V	
,DSTYPE=M	
,DSTYPE=T	

,FILESEQ=*number*	*number*: 1-9999
,FILESEQ=*reg*	*reg*: register (2) - (12)
,GENERIC=YES	
,GENERIC=ASIS	**Default:** GENERIC=ASIS
,GROUPID=*'groupid'*	*groupid*: 1- to 8-character group ID
,GROUPID=*groupid addr*	*groupid addr*: A-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12)
,LOG=ASIS	**Default:** LOG=ASIS
,LOG=NOFAIL	
,LOG=NONE	
,LOG=NOSTAT	
,LOGSTR=*logstr addr*	*logstr addr*: A-type address or register (2) - (12)
,OLDVOL=*old vol addr*	*old vol addr*: A-type address or register (2) - (12)
,RACFIND=YES	
,RACFIND=NO	
,RECVR=*recvr addr*	*recvr addr*: A-type address or register (2) - (12)
,RELEASE=*number*	*number*: 1.9.2, 1.9, 1.8X, 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,RTOKEN=*rtoken addr*	*rtoken addr*: A-type address or register (2) - (12)
,STATUS=NONE	**Default:** STATUS=NONE
,STATUS=EVERDOM	
,STATUS=WRITEONLY	
,STATUS=ACCESS	
,TAPELBL=STD	**Default:** TAPELBL=STD

,TAPELBL=BLP

,TAPELBL=NL

,USERID=*'userid'* *userid*: 1- to 8-character user ID

,USERID=*userid addr* *userid addr*: A-type address or register (2) - (12)

,UTOKEN=*token addr* *token addr*: A-type address or register (2) - (12)

,MF=S

The parameters are explained as follows:

,ACCLVL=*access level addr*
,ACCLVL=(*access level addr,parm list addr*)
specifies the tape-label access-level information for the z/OS tape-label functions. The access level pointed to by the specified address is a 1-byte length field, containing the value (0-8) of the length of the following data, followed by an 8-character string that will be passed to the RACHECK installation-exit routines. The optional parameter list pointed to by the specified address contains additional information to be passed to the RACHECK installation-exit routines. RACF does not inspect or modify this information.

On z/VM, the address must point to a 1-byte length field, followed by the parameter list. Note that the parameter list must not contain any addresses.

,ACEE=*acee addr*
specifies the address of the ACEE to be used during RACF authorization-check processing.

If no ACEE is specified, RACF uses the TASK ACEE pointer (TCBSENV) in the extended task control block (TCB). Otherwise, or if the TASK ACEE pointer is zero, RACF uses the main ACEE for the address space. The main ACEE is pointed to by the ASXBSENV field of the address-space extension block.

On z/VM, the ACEE used should have been created as the result of an earlier RACROUTE invocation (for example, REQUEST=VERIFY,ENVIR=CREATE).

When RACF is installed, the caller of RACROUTE REQUEST=AUTH must have at least UPDATE authority to the ICHCONN profile in the FACILITY class to use the USERID and ACEE keywords. For details on the ICHCONN profile, see *z/VM: RACF Security Server Security Administrator's Guide*.

,APPL=*'applname'*
,APPL=*applname addr*
specifies the name of the application requesting authorization checking. The *application name* is not used for the authorization checking process but is made available to the installation exit routine or routines called by the RACROUTE REQUEST=AUTH routine. If the address is specified, the address must point to an 8-byte field containing the application name, left-justified and padded with blanks.

,ATTR=READ
,ATTR=UPDATE
,ATTR=CONTROL
,ATTR=ALTER
,ATTR=*reg*
specifies the level of authority requested. RACF checks the resource profile protecting the resource identified by the ENTITY and CLASS keywords. The values have the following hierarchical order:

READ

UPDATE

CONTROL

ALTER.

That is, if a user has update authority and ATTR=READ is specified, RACF returns a return code of 0. If ATTR=CONTROL, RACF returns a return code of 8.

If a register is specified, the register must contain one of the following codes in the low-order byte of the register:

X'02' READ
X'04' UPDATE
X'08' CONTROL
X'80' ALTER.

The default is ATTR=READ.

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies that RACF authorization checking is to be performed for a resource of the specified class. The address must point to a 1-byte field indicating the length of the class name, followed by the class name.

The specified class must be defined in the RACF class descriptor table, and must be active for this request to be processed. In addition, if the class descriptor table specifies that RACLIST is required, the SETROPTS RACLIST option must be active for the class.

,DSTYPE=N
,DSTYPE=V
,DSTYPE=M
,DSTYPE=T
specifies the type of data set associated with the request:

N
for non-VSAM

V
for VSAM

M
for model profile

T
for tape.

If DSTYPE=T is specified and tape data-set protection is not active, the processing is the same as for RACROUTE REQUEST=AUTH CLASS=TAPEVOL.

DSTYPE should be specified only for CLASS=DATASET.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,ENTITY=*resource name addr*
,ENTITY=(*resource name addr*)
,ENTITYX=*extended resource name addr*
,ENTITYX=(*extended resource name addr*)
specifies the resource address.

Consideration:
IBM recommends that you use ENTITYX rather than ENTITY. With ENTITY, the entity name you pass to RACF must be in a buffer, the size of which is determined by the length in the RACF class-descriptor table (CDT). If the maximum length of a class-descriptor entity increases in the future, you must modify your program to use a larger buffer. By using ENTITYX, you avoid this possible problem because you remove the CDT dependency from your program.

For the ENTITY keyword, the resource name is a 44-byte DASD data-set name for CLASS=DATASET, or a 6-byte volume serial number for CLASS=DASDVOL or CLASS=TAPEVOL. The length of all other resource names is determined from the class-descriptor tables.

- ENTITY=*resource name addr* or ENTITY=(*resource name addr*) specifies that RACF authorization checking is to be performed for the resource whose name is pointed to by the specified address. The name must be left-justified in the field and padded with blanks.
- ENTITYX=*extended resource address* or ENTITYX=(*extended resource address*) specifies the address of a structure that consists of two 2-byte length fields, followed by the entity name.
  - The first 2-byte field specifies a buffer length, which can be from 0 to 255 bytes. This length field refers to the length of the buffer that contains the entity name; it does not include the length of either length field.
  - The second 2-byte field specifies the actual length of the entity name. This length field includes the length of the actual name without any trailing blanks; it does not include the length of either length field.

  These two length fields can be used in several different ways:

  - If you know the length of the entity name, you can specify 0 in the first field and the length of the entity name in the second field. When you specify the second field, note that each byte counts. This means the entity name that you specify must match exactly the entity name on the RACF database.
  - If you choose to place the entity name in a buffer area, specify the length of the buffer in the first field. For the second field, do one of the following:
    - If you know the length of the entity name, specify the length in the second field. The length of the first field can be from 0 to 255, but *must* be equal to or greater than the length of the second field.
    - If you do not know the length of the entity name, specify 0 in the second field, and RACF will count the number of characters in the entity name.

  To use this keyword, you must also specify RELEASE=1.9 or later.

,FILESEQ=*number*
,FILESEQ=*reg*
specifies the file-sequence number of a tape data set on a tape volume or within a tape-volume set. The value must be in the range 1 - 9999. If a register is specified, it must contain the file-sequence number in the low-order halfword.

If CLASS=DATASET and DSTYPE=T are not specified, FILESEQ is ignored.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

,GENERIC=YES
,GENERIC=ASIS
specifies whether the resource name is to be treated as a generic profile name. GENERIC is ignored if the GENCMD option on the RACF SETROPTS command is not specified for the class (see *z/VM: RACF Security Server Command Language Reference*).

This keyword is designed primarily for use by RACF commands.

YES
The resource name is considered a generic profile name, even if it does not contain a generic character: an asterisk (*), a percent sign (%), or, for general-resource classes, an ampersand sign (&).

ASIS
The resource name is considered generic if it contains a generic character: an asterisk (*), a percent sign (%), or, for general-resource classes, an ampersand sign (&).

,GROUPID=*'groupid'*
,GROUPID=*groupid address*
specifies the group ID that RACF uses to perform third-party authorization checking. This is an 8-character field, left-justified, and padded to the right with blanks.

If the calling program wants a third-party authorization check performed on the group ID rather than the user ID, the USERID keyword must be specified as *NONE*. That is, when the caller invokes third-party authorization checking, RACF verifies the authority of the group ID to the requested resource; RACF disregards the group ID associated with the ACEE of the caller.

,INSTLN=*parm list addr*
specifies the address of an area that is to contain parameter information meaningful to the RACHECK installation exit routine. This information is passed to the installation exit routine when it is given control by RACROUTE REQUEST=AUTH.

The INSTLN parameter can be used by an application program acting as a resource manager that needs to pass information to the RACHECK installation exit routine.

The address must point to a 1-byte length field, followed by the parameter list. The parameter list must not contain any addresses.

,LOG=ASIS
,LOG=NOFAIL
,LOG=NONE
,LOG=NOSTAT
specifies the types of access attempts to be recorded on the SMF DATA file for z/VM:

ASIS
RACF records the event in the manner specified in the profile that protects the resource, or by other methods such as a SETROPTS option.

NOFAIL
If the authorization check fails, the attempt is not recorded. If the authorization check succeeds, the attempt is recorded as in ASIS.

NONE
The attempt is not recorded.

NOSTAT
The attempt is not recorded. No logging occurs and no resource statistics (including messages and SMF records) are updated.

,LOGSTR=*logstr addr*
specifies a variable-length data string consisting of a 1-byte, binary length field followed by character data that is to be included in the RACF SMF process records. The character data can be 0 to 255 bytes long. The RACF report writer includes LOGSTR data on the process reports.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

,OLDVOL=*old vol addr*
specifies a volume serial number:

- For CLASS=DATASET, within the same multivolume data set specified by VOLSER=
- For CLASS=TAPEVOL, within the same tape volume specified by ENTITY=.

RACF authorization checking verifies that the OLDVOL specified is part of the same multivolume data set or tape-volume set. RACF authorization checking will not look at global access table entries when the OLDVOL parameter is specified.

The specified address points to the field that contains the volume serial number padded to the right with blanks, if necessary, to make 6 characters.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,RACFIND=YES
,RACFIND=NO
indicates whether the resource is meant to be protected by a discrete profile. The RACF processing and the possible return codes are given in Table 3 on page 42.

Note: In all cases, a return code of X'0C' is also possible if the OLDVOL specified was not part of the multivolume data set defined by VOLSER, or if it was not part of the same tape volume defined by ENTITY.

Table 3. Types of Profile Checking Performed by RACROUTE REQUEST=AUTH

Operand	Generic Profile Checking Inactive	Generic Profile Checking Active
RACFIND=YES	Look for discrete profile; if found, exit with return code 00 or 08. If no discrete profile is found, exit with return code 08.	Look for discrete profile; if found, exit with return code 00 or 08. Look for generic profile; if found, exit with return code 00 or 08. Exit with return code 08 if neither a discrete nor a generic profile is found.
RACFIND=NO	No checking. Exit with return code 04.	Look for generic profile; if found, exit with return code 00 or 08. If not found, exit with return code 04.
RACFIND not specified	Look for discrete profile; if found, exit with return code 00 or 08. If no discrete profile is found, exit with return code 04.	Look for discrete profile; if found, exit with return code 00 or 08. Look for generic profile; if found, exit with return code 00 or 08. Exit with return code 04 if neither a discrete nor a generic profile is found.

,RECVR=*recvr addr*
specifies the address of the user ID that has the authority to access the resource if a resource profile does not exist to protect it. The field is 8 bytes, left-justified and padded to the right with blanks.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=AUTH macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

,RTOKEN=*rtoken addr*
specifies the address of the RTOKEN of a unit of work. The first byte contains the length of the RTOKEN, followed by the UTOKEN of the creator of the resource. See the explanation of UTOKEN.

To use this keyword, you must also specify RELEASE=1.9 or later.

,STATUS=NONE
,STATUS=EVERDOM
,STATUS=WRITEONLY
,STATUS=ACCESS
specifies the type of status required.

NONE
No STATUS= functions have been requested.

EVERDOM
Security-label authorization checking includes a check to see whether the user has a security label, other than that of this job or logon session, that could *ever* dominate that of the current object. This is done primarily so that message security can determine what to do with the messages that cannot currently be shown to the user. For example, if the user does not have a security label that can ever dominate that of the message, the message may be deleted. There are no restrictions on the CLASS parameter. Be aware that choosing this option increases processing time. The default is that security-label authorization checking occurs with the security label of the current job or logon session.

WRITEONLY
The request is for output only in a class that also allows read or write functions. No reading is to be done.

ACCESS
The request is simply to return the user's highest current access to the resource specified. Upon successful completion, the user's access is returned in the RACF reason code. No auditing is done for this request.

Note: If the ATTR= keyword is specified along with STATUS=ACCESS, the ATTR= keyword will be ignored.

,TAPELBL=STD
,TAPELBL=BLP
,TAPELBL=NL
specifies the type of tape-label processing to be done:

STD
IBM or ANSI standard labels

BLP
Bypass label processing

NL
Non-labeled tapes.

For TAPELBL=BLP, the user must have the requested authority to the profile ICHBLP in the general-resource class FACILITY. For more information about using the ICHBLP profile on z/OS, see *z/VM: RACF Security Server Security Administrator's Guide*.

On z/OS for TAPELBL=NL or BLP, data management routines will not allow the user to protect volumes with volume serial number in the format "*Lnnnnn*."

This parameter is primarily intended for use by data-management routines to indicate the label type from the LABEL keyword on the JCL statement.

This parameter is valid for CLASS=DATASET and DSTYPE=T, or CLASS=TAPEVOL.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

,USERID=*'userid'*
,USERID=*userid address*
specifies the userID that RACF uses to perform third-party authorization checking. This is an 8-character field that is left-justified and padded to the right with blanks.

If USERID is specified when the caller invokes RACROUTE REQUEST=AUTH, RACF verifies that user's authority to the given entity; RACF disregards the user ID associated with the ACEE of the caller.

Note: If the calling program does not specify the GROUPID keyword, the internal RACROUTE REQUEST=VERIFY function uses the default group associated with the specified user ID.

When RACF is installed, the caller of RACROUTE REQUEST=AUTH must have at least UPDATE authority to the ICHCONN profile in the FACILITY class to use the USERID and ACEE keywords. For details on the ICHCONN profile, see *z/VM: RACF Security Server Security Administrator's Guide*.

,UTOKEN=*token addr*
specifies the address of the UTOKEN of the user for whom RACF will perform third-party authorization checking. The first byte contains the length of the UTOKEN, and the second byte contains the version number.

If UTOKEN is specified when the caller invokes RACROUTE REQUEST=AUTH, RACF verifies that user's authority to the given entity; RACF disregards the user ID associated with the ACEE of the caller.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

,VOLSER=*vol addr*
specifies the volume serial number, as follows:

- For non-VSAM DASD data sets and tape data sets, this is the volume serial number of the volume on which the data set resides.
- For VSAM DASD data sets, this is the volume serial number of the catalog controlling the data set.

The volume serial number is optional if DSTYPE=M is specified; it is ignored if the profile name is generic.

The field pointed to by the specified address contains the volume serial number, padded to the right with blanks if necessary to make six characters. VOLSER= is only valid (and must be supplied) with CLASS=DATASET, (unless DSTYPE=M is specified) when ENTITY or ENTITYX is also coded.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,MF=S
specifies the standard form of the RACROUTE REQUEST=AUTH macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note: Some RACROUTE parameter errors result in a RACF abend. On z/VM, these abends are simulated by RACF return and reason codes. For RACROUTE REQUEST=AUTH, RACF return code 282 corresponds to a RACF abend that is documented in *z/VM: RACF Security Server Messages and Codes*. The reason code also reflects the abend reason code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=AUTH completed successfully.

RACF Return Code
Meaning

00
The user is authorized by RACF to obtain use of a RACF-protected resource.

Reason Code
Meaning

00
Indicates a normal completion.

04
Indicates the warning status of the resource was requested by the RACROUTE REQUEST=AUTH issuer's setting bit X'10' at offset 12 decimal in the request-specific portion of the RACROUTE REQUEST=AUTH parameter list with the resource in warning mode.
The request-specific portion of the RACROUTE REQUEST=AUTH parameter list follows the RACROUTE parameter list (ICHSAFP) and is mapped by mapping macro, ICHACHKL.

10
When CLASS=TAPEVOL, indicates the TAPEVOL profile contains a TVTOC.

20
When CLASS=TAPEVOL, indicates that the TAPEVOL profile can contain a TVTOC, but currently does not (for a scratch pool volume).

24
When CLASS=TAPEVOL, indicates that the TAPEVOL profile does not contain a TVTOC.

14
Requested function with STATUS=ACCESS specified has completed successfully. The user's highest access to the specified resource is indicated by one of the following reason codes:

Reason Code
Meaning

00
The user has no access.

04
The user has READ authority.

08
The user has UPDATE authority.

0C
The user has CONTROL authority.

10
The user has ALTER authority.

04
Requested function could not be completed. No RACF decision.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
One of the following has occurred:

- RACF is not installed
- Specified requester, subsystem, or class is not in the RACF router table
- Specified class is not in the RACF class descriptor table.

04
The specified resource is not protected by RACF.

Reason Code
Meaning

00
One of the following has occurred:

- There is no RACF profile protecting the resource
- RACF is not active
- Specified class is not active
- Specified class requires SETROPTS RACLIST option to be active and it is not.

582
Reserved.

08
Requested function has failed.

RACF Return Code
Meaning

08
The user is *not* authorized by RACF to obtain use of the specified RACF-protected resource.

Reason Code
Meaning

00
Indicates a normal completion.

08
Indicates DSTYPE=T or CLASS=TAPEVOL was specified and the user is not authorized to use the specified volume.

0C
Indicates the user is not authorized to use the data set.

10
Indicates DSTYPE=T or CLASS=TAPEVOL was specified and the user is not authorized to specify TAPELBL=(,BLP).

14
Indicates the user is not authorized to open a noncataloged data set.

18
Indicates the user is not authorized to issue RACROUTE REQUEST=AUTH when system is in tranquil state (MLQUIET).

20
The user's security label does not dominate that of the resource; it fails SECLABEL authorization checking.

24
The user's security label can never dominate that of the resource.

28
The resource must have a security label, but does not have one.

0C
The OLDVOL specified was not part of the multivolume data set defined by VOLSER, or it was not part of the same tape volume defined by ENTITY.

10
RACROUTE REQUEST=VERIFY was issued by a third party, and RACROUTE REQUEST=AUTH failed.

Reason Code
Meaning

XXXX
Refer to "RACROUTE REQUEST=VERIFY: Identify and Verify a RACF-Defined User" on page 162 for the explanation of this reason code. Under SAF return code X'08', see RACF return code *XXXX*.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=AUTH macro; however, the list form of the macro does not have the same RELEASE parameter. Macro processing terminates.

CDT Default Return Codes and Reason Codes

Normally, if a resource profile is not found, the function returns a return code of 4. However, beginning with RACF 1.9, if a resource profile is not found, *but* a default return-code keyword is specified in the CDT for the class specified on the RACROUTE REQUEST=AUTH, the function returns that specified return code.

When a default return code of other than 4 is specified for a class in the CDT, that specified return code is returned and the reason code is incremented by hexadecimal 200.

Example

Operation: Perform third-party RACF authorization checking, using the standard form, for a minidisk with a 4-character virtual address on a z/VM system. RACF does not allow the first character of a 4-character virtual address to be zero (0). For example, RACF allows SMITH.191, SMITH.1234, and SMITH.002, but not SMITH.0191. Use the following RACROUTE request to request an authority check on Smith's A-disk, which on a z/VM system would have a virtual address of 0191.

```
RACROUTE REQUEST=AUTH,CLASS=CLASSNL,ENTITY=ENTITYNA,       X
         RELEASE=1.9.2,MF=S,WORKA=WORK,ATTR=READ,          X
         USERID=IBMUSER
   .
   .
   .
ENTITYNA  DC  CL39'SMITH.191'     * entity
CLASSNL   DC  XL1'07'             * class name length
CLASSN    DC  CL8'VMMDISK'        * class name
IBMUSER   DC  CL8'SUE'            * requesting user ID
          DS  0D                  * ensure doubleword alignment
WORK      DS  CL512               * storage for macro expansion
```

For further details on how to protect z/VM minidisks, see the *z/VM: RACF Security Server Security Administrator's Guide*.

RACROUTE REQUEST=AUTH (List Form)

The list form of the RACROUTE REQUEST=AUTH macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=AUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=AUTH	
,ACCLVL=*access level addr*	*access level addr*: A-type address

,ACEE=*acee addr*	*acee addr*: A-type address
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address
,ATTR=READ	**Default:** ATTR=READ
,ATTR=UPDATE	
,ATTR=CONTROL	
,ATTR=ALTER	
,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address
,DSTYPE=N	**Default:** DSTYPE=N
,DSTYPE=V	
,DSTYPE=M	
,DSTYPE=T	
,ENTITY=(*resource name addr*)	*resource name addr*: A-type address
,ENTITYX=*extended resource name addr*	*extended resource name addr*: A-type address
,FILESEQ=*number*	*number*: 1-9999
,GENERIC=YES	
,GENERIC=ASIS	**Default:** GENERIC=ASIS
,GROUPID=*'groupid'*	*groupid*: 1- to 8-character group ID
,GROUPID=*groupid addr*	*groupid addr*: A-type address
,INSTLN=*parm list addr*	*parm list addr*: A-type address
,LOG=ASIS	**Default:** LOG=ASIS
,LOG=NOFAIL	
,LOG=NONE	
,LOG=NOSTAT	

,LOGSTR=*logstr addr*	*logstr addr*: A-type address
,OLDVOL=*old vol addr*	*old vol addr*: A-type address
,RACFIND=YES ,RACFIND=NO	
,RECVR=*recvr addr*	*recvr addr*: A-type address
,RELEASE=*number*	*number*: See Standard Form **Default:** RELEASE=1.6
,RTOKEN=*rtoken addr*	*rtoken addr*: A-type address
,STATUS=NONE ,STATUS=EVERDOM ,STATUS=WRITEONLY ,STATUS=ACCESS	**Default:** STATUS=NONE
,TAPELBL=STD ,TAPELBL=BLP ,TAPELBL=NL	**Default:** TAPELBL=STD
,USERID=*'userid'* ,USERID=*userid addr*	*userid*: 1- to 8-character user ID *userid addr*: A-type address
,UTOKEN=*token addr*	*token addr*: A-type address
,VOLSER=*vol addr*	*vol addr*: A-type address **Note:** VOLSER is required on either the list or the execute form of the macro for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=AUTH macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=AUTH macro instruction.

RACROUTE REQUEST=AUTH (Execute Form)

The execute form of the RACROUTE REQUEST=AUTH macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=AUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=AUTH	
,ACCLVL=*access level addr*	*access level addr*: Rx-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER ,ATTR=*reg*	*reg*: Register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	
,ENTITY=(*resource name addr*)	*resource name addr*: Rx-type address or register (2) - (12)

,ENTITYX=*extended resource name addr*
extended resource name addr: Rx-type address or register (2) - (12)

,FILESEQ=*number*
number: 1-9999

,FILESEQ=*reg*
reg: Register (2) - (12)

,GENERIC=YES

,GENERIC=ASIS

,GROUPID=*groupid addr*
groupid addr: Rx-type address or register (2) - (12)

,INSTLN=*parm list addr*
parm list addr: Rx-type address or register (2) - (12)

,LOG=ASIS

,LOG=NOFAIL

,LOG=NONE

,LOG=NOSTAT

,LOGSTR=*logstr addr*
logstr addr: Rx-type address or register (2) - (12)

,OLDVOL=*old vol addr*
old vol addr: Rx-type address or register (2) - (12)

,RACFIND=YES

,RACFIND=NO

,RECVR=*recvr addr*
recvr addr: Rx-type address or register (2) - (12)

,RELEASE=*number*
number: See Standard Form

,RELEASE=(,CHECK)
Default: RELEASE=1.6

,RELEASE=(*number*,CHECK)

,RTOKEN=*rtoken addr*
rtoken addr: Rx-type address or register (2) - (12)

,STATUS=NONE

,STATUS=EVERDOM

,STATUS=WRITEONLY

,STATUS=ACCESS

,TAPELBL=STD

,TAPELBL=BLP

,TAPELBL=NL

,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,UTOKEN=*token addr*	*token addr*: Rx-type address or register (2) - (12)
,VOLSER=*vol addr*	*vol addr*: Rx-type address or register (2) - (12) **Note:** VOLSER is required on either the list or the execute form of the macro for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address, or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=AUTH macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=AUTH macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=AUTH macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=AUTH macro instruction.

RACROUTE REQUEST=AUTH (Modify Form)

The modify form of the RACROUTE REQUEST=AUTH macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=AUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=AUTH	
,ACCLVL=*access level addr*	*access level addr*: Rx-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER ,ATTR=reg	*reg*: Register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	
,ENTITY=(*resource name addr*)	*resource name addr*: Rx-type address or register (2) - (12)
,ENTITYX=*extended resource name addr*	*extended resource name addr*: Rx-type address or register (2) - (12)

,FILESEQ=*number* — *number*: 1-9999

,FILESEQ=*reg* — *reg*: Register (2) - (12)

,GENERIC=YES

,GENERIC=ASIS

,GROUPID=*groupid addr* — *groupid addr*: Rx-type address or register (2) - (12)

,INSTLN=*parm list addr* — *parm list addr*: Rx-type address or register (2) - (12)

,LOG=ASIS

,LOG=NOFAIL

,LOG=NONE

,LOG=NOSTAT

,LOGSTR=*logstr addr* — *logstr addr*: Rx-type address or register (2) - (12)

,OLDVOL=*old vol addr* — *old vol addr*: Rx-type address or register (2) - (12)

,RACFIND=YES

,RACFIND=NO

,RECVR=*recvr addr* — *recvr addr*: Rx-type address or register (2) - (12)

,RELEASE=*number* — *number*: See Standard Form

,RELEASE=(,CHECK) — **Default:** RELEASE=1.6

,RELEASE=(*number*,CHECK)

,RTOKEN=*rtoken addr* — *rtoken addr*: Rx-type address or register (2) - (12)

,STATUS=NONE

,STATUS=EVERDOM

,STATUS=WRITEONLY

,STATUS=ACCESS

,TAPELBL=STD

,TAPELBL=BLP

,TAPELBL=NL	
,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,UTOKEN=*token addr*	*token addr*: Rx-type address or register (2) - (12)
,VOLSER=*vol addr*	*vol addr*: Rx-type address or register (2) - (12) **Note:** VOLSER is required on either the list or the execute form of the macro for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address, or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=AUTH macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=AUTH macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=AUTH macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=AUTH macro instruction.

RACROUTE REQUEST=DEFINE: Define, Modify, Rename, or Delete a Resource for RACF

The RACROUTE REQUEST=DEFINE macro defines, modifies, renames, or deletes resource profiles for RACF. You can also use it for special cases of authorization checking. RACF uses the resulting profiles to perform authorization checking when a user requests access to a RACF-protected resource.

The RACDEF preprocessing and postprocessing exit routines can change or add the RACROUTE REQUEST=DEFINE parameters OWNER, LEVEL, UACC, or AUDIT.

When RACF is installed, the caller of RACROUTE REQUEST=DEFINE must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

RACROUTE REQUEST=DEFINE (Standard Form)

The standard form of the RACROUTE REQUEST=DEFINE macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

Note:

RACROUTE REQUEST=DEFINE requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=DEFINE.

The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=DEFINE	
,ENTITY=*profile name addr*	*profile name addr*: A-type address or register (2) - (12)
,ENTITYX=*extended profile name addr*	*extended profile name addr*: A-type address or register (2) - (12)
,VOLSER=*vol addr*	*vol addr*: A-type address or register (2) - (12) **Note:** VOLSER is required for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,ACCLVL=*access value addr*	*access value addr*: A-type address or register (2) - (12)
,ACCLVL=(*access value addr,parm list addr*)	*parm list addr*: A-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,AUDIT=NONE	**Note:** AUDIT is valid if TYPE=DEFINE is specified.
,AUDIT=*audit value*	*audit value*: ALL, SUCCESS, or FAILURES
,AUDIT=(*audit value*(*access level*),*audit value*(*access level*))	**Default:** AUDIT=READ
,AUDIT=*reg*	*reg*: register (2) - (12)

,CHKAUTH=YES ,CHKAUTH=NO	**Default:** CHKAUTH=NO
,CLASS=*'class name'* ,CLASS=*class name addr*	*class name*: 1- to 8-character name. *class name addr*: A-type address or register (2) - (12) **Default:** CLASS=DATASET
,DATA=*data addr*	*data addr*: A-type address or register (2) - (12)
,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	**Default:** DSTYPE=N
,ENVIR=VERIFY	Specifies that only verification is to be done. **Default:** Normal RACROUTE REQUEST=DEFINE processing
,ERASE=YES ,ERASE=NO	**Default:** ERASE=NO
,EXPDT=*expir-date addr* ,EXPDTX=*extended expir-date addr*	*expir-date addr*: A-type address or register (2) - (12) *extended expir-date addr*: A-type address or register (2) - (12)
,RETPD=*retn-period addr*	*retn-period addr*: A-type address or register (2) - (12) **Default:** See description of parameter.
,FILESEQ=*number* ,FILESEQ=*reg*	*number*: 1-9999 *reg*: Register (2) - (12)
,GENERIC=YES ,GENERIC=ASIS	**Default:** GENERIC=ASIS
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12)
,LEVEL=*number* ,LEVEL=*reg*	**Default:** LEVEL=zero *reg*: Register (2) - (12)
,MCLASS=*'class name'* ,MCLASS=*class name addr*	*class name*: 1- to 8-character name *class name addr*: A-type address or register (2) - (12)

	Default: MCLASS=DATASET
,MENTITY=*entity addr* ,MENTX=*extended entity addr*	*entity addr*: A-type address or register (2) - (12) *extended entity addr*: A-type address or register (2) - (12)
,MGENER=ASIS ,MGENER=YES	**Default:** MGENER=ASIS
,MGMTCLA=*management type addr*	*management type addr*: A-type address or register (2) - (12)
,MVOLSER=*volser addr*	*volser addr*: A-type address or register (2) - (12)
,NOTIFY=*notify-id addr*	*notify-id addr*: A-type address or register (2) - (12)
,OWNER=*owner id addr*	*owner id addr*: A-type address or register (2) - (12)
,RACFIND=YES ,RACFIND=NO	
,RELEASE=*number*	*number*: 1.9.2, 1.9, 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,RESOWN=*resource owner addr*	*resource owner addr*: A-type address or register (2) - (12)
,SECLABL=*addr*	*addr*: A-type address or register (2) - (12)
,SECLVL=*addr*	*addr*: A-type address or register (2) - (12)
,STORCLA=*storage class addr*	*storage class addr*: A-type address or register (2) - (12)
,TAPELBL=STD ,TAPELBL=BLP ,TAPELBL=NL	**Default:** TAPELBL=STD
,TYPE=DEFINE	**Default:** TYPE=DEFINE
,TYPE=DEFINE,NEWNAME =*new resource name addr*	*new resource name addr*: A-type address or register (2) - (12)

,TYPE=DEFINE,NEWNAMX =*extended new resource name addr*	*extended new resource name addr*: A-type address or register (2) - (12)
,TYPE=ADDVOL,OLDVOL =*old vol addr*	*old vol addr*: A-type address or register (2) - (12)
,TYPE=CHGVOL,OLDVOL =*old vol addr*	
,TYPE=DELETE	
,UACC=ALTER	
,UACC=CONTROL	
,UACC=UPDATE	
,UACC=READ	
,UACC=NONE	
,UACC=*reg*	*reg*: Register (2) - (12)
,UNIT=*unit addr*	*unit addr*: A-type address or register (2) - (12)
,WARNING=YES	
,WARNING=NO	**Default:** WARNING=NO **Note:** WARNING is valid if TYPE=DEFINE is specified.
,MF=S	

The parameters are explained as follows:

,ENTITY=*profile name addr*
,ENTITYX=*extended profile name addr*
specifies the address:

- ENTITY=*profile name addr* specifies the address of the name of the discrete or generic profile that is to be defined to, modified, or deleted from RACF. The profile name is a 44-byte DASD data-set name for CLASS=DATASET, or a 6-byte volume serial name for CLASS=DASDVOL or CLASS=TAPEVOL. The lengths of all other profile names are determined by the class-descriptor table. The name must be left-justified in the field and padded with blanks.
- ENTITYX=*extended profile name addr* specifies the address of a structure that consists of two 2-byte length fields, followed by the entity name.
  - The first 2-byte field specifies a buffer length which can be from 0 to 255 bytes. This length field refers to the length of the buffer that contains the entity name; it does not include the length of either length field.
  - The second 2-byte field specifies the actual length of the entity name. This length field includes the length of the actual name without any trailing blanks; it does not include the length of either length field.

  These two length fields can be used in several different ways:

- If you know the length of the entity name, you can specify 0 in the first field and the length of the entity name in the second field. When you specify the second field, note that each byte counts. This means the entity name that you specify must match exactly the entity name on the RACF database.
- If you choose to place the entity name in a buffer area, specify the length of the buffer in the first field. For the second field, do one of the following:
  - If you know the length of the entity name, specify the length in the second field. The length of the first field can be from 0 to 255, but *must* be equal to or greater than the length of the second field.
  - If you do not know the length of the entity name, specify 0 in the second field, and have RACF determine the number of characters in the entity name.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

Consideration:
IBM recommends that you use ENTITYX rather than ENTITY. With ENTITY, the entity name you pass to RACF must be in a buffer, the size of which is determined by the length in the RACF class-descriptor table (CDT). If the maximum length of a class-descriptor entity increases in the future, you must modify your program to use a larger buffer. By using ENTITYX, you avoid this possible problem because you have removed the CDT dependency from your program.

,VOLSER=*vol addr*
specifies the address of the volume serial number:

- For TYPE=ADDVOL, of the new volume to be added to the definition of the data set
- For TYPE=ADDVOL and CLASS=TAPEVOL, of the new volume being added to the tape volume set identified by ENTITY or ENTITYX
- For TYPE=DEFINE and CLASS=DATASET, of the catalog (for a VSAM data set), or of the volume on which the data set resides (for a non-VSAM data set).

The volume serial number is optional if DSTYPE=M is specified; it is ignored if the profile name is generic.

The field pointed to by the specified address contains the volume serial number, padded to the right with blanks if necessary to make six characters.

z/On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,ACCLVL=*access value addr*
,ACCLVL=(*access value addr,parm list addr*)
specifies the tape-label access-level information for the z/OS tape-label functions. The address must point to a field containing a one-byte length field (with a value that can range from 0 to 8) followed by an 8-character string that will be passed to the RACDEF installation exit routines. The parameter list address points to a parameter list containing additional information to be passed to the RACDEF installation exit routines.

RACF does not check or modify this information.

,ACEE=*acee addr*
specifies the address of the ACEE to be used during RACROUTE REQUEST=DEFINE processing.

The ACEE should have been created as the result of a previous RACROUTE invocation (such as REQUEST=VERIFY,ENVIR=CREATE).

,AUDIT=NONE
,AUDIT=*audit value*
,AUDIT=(*audit value*(*access level*),*audit value*(*access level*),. . .)
,AUDIT=*reg*
specifies the types of accesses and the access levels that are to be logged to the SMF data file.

For *audit value*, specify one of the following: ALL, SUCCESS, or FAILURES. You may optionally specify an *access level*(*access authority*) following each *audit value.*

Access Levels:

- READ logs access attempts at any level. READ is the default access-level value.
- UPDATE logs access attempts at the UPDATE, CONTROL, and ALTER levels.
- CONTROL logs access attempts at the CONTROL and ALTER levels.
- ALTER logs access attempts at the ALTER level only.

Note: For more information about specific audit values and access levels, see *z/VM: RACF Security Server Command Language Reference*.

RACF resolves combinations of conflicting specifications by using the most encompassing specification. Thus, in the case of the following:

```
ALL(UPDATE),FAILURES(READ)
```

RACF assumes SUCCESS(UPDATE),FAILURES(READ).

For compatibility with previous releases, register notation can also be specified as AUDIT=*reg* if the register is not given as a symbolic name; for example, ALL, SUCCESS, or FAILURES.

Logging is controlled separately for SUCCESS and FAILURES, and can also be suppressed or requested using the RACHECK postprocessing installation exit routine.

If a register is specified, its low-order byte must contain one of the following valid audit values:

Bit
Meaning

0
ALL

1
SUCCESS

2
FAILURES

3
NONE

4-5
Qualifier for SUCCESS

6-7
Qualifier for FAILURES

The qualifier codes are as follows:

00
READ

01
UPDATE

10
CONTROL

11
ALTER

Only one of bits 0 through 3 can be on. If ALL is specified, the two qualifier fields can be used to request different logging levels for successful and unsuccessful events.

Note: RACF does not check the validity of the audit type if it has been added or modified by the RACDEF preprocessing or postprocessing exit routine.

AUDIT is valid if TYPE=DEFINE is specified.

,CHKAUTH=YES
,CHKAUTH=NO
specifies whether or not an internal RACROUTE REQUEST=AUTH with ATTR=ALTER is to be done to verify that the user is authorized to perform the operation.

CHKAUTH=YES is valid when TYPE=DEFINE,NEWNAME or TYPE=DEFINE,NEWNAMX, or TYPE=DELETE is specified.

For DSTYPE=T, CHKAUTH=YES specifies that an internal RACROUTE REQUEST=AUTH with ATTR=UPDATE be done to verify that the user is authorized to define a data set (TYPE=DEFINE), delete a data set (TYPE=DELETE), or add a volume (TYPE=ADDVOL).

The default is CHKAUTH=NO.

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies that a profile is to be defined, modified, or deleted in the specified class. If an address is specified, the address must point to a one-byte length field followed by the class name (such as DATASET or TAPEVOL). The class name should be no longer than 8 characters.

,DATA=*data addr*
specifies the address of a field that contains up to 255 characters of installation-defined data to be placed in the profile. The data address must point to a field containing a one-byte length field (whose value can range from 0 to 255) followed by the actual installation-defined data.

DATA is valid if TYPE=DEFINE is specified.

,DSTYPE=N
,DSTYPE=V
,DSTYPE=M
,DSTYPE=T
specifies the type of data set associated with the request:

N
for non-VSAM

V
for VSAM

M
for model profile

T
for tape.

If DSTYPE=T is specified and tape data-set protection is not active, the processing is the same as for RACROUTE REQUEST=DEFINE, CLASS='TAPEVOL'.

Specify DSTYPE only when the value of CLASS is DATASET.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,ENVIR=VERIFY
specifies that no profile is to be created, but that the user's authority to define or rename the resource or profile is to be checked, along with any other authorization processing that is necessary.

If you specify ENVIR, you must also specify RELEASE=1.8.1 or a later release number.

Note: If you do not specify ENVIR=VERIFY, normal RACROUTE REQUEST=DEFINE processing occurs.

,ERASE=YES
,ERASE=NO
specifies whether the DASD data set, or the released space, is to be erased when it is deleted or part of its space is to be released for reuse.

If ERASE=YES is specified, the data set is erased when it is deleted or released for reuse.

If ERASE=NO is specified, the data set is not erased, deleted, or released.

The default is ERASE=NO.

Specify ERASE only for CLASS=DATASET.

Note: This parameter may be overridden by the RACF SETROPTS ERASE command.

On z/VM, data sets may exist on OS or DOS minidisks, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,EXPDT=*expir-date addr*
,EXPDTX=*extended expir-date addr*
,RETPD=*retn-period addr*
specifies the address containing information about the expiration date or RACF security retention period of the data set.

- EXPDT=*expir-date addr* specifies the address of a 3-byte field containing the expiration date of the data set. The date is given in packed decimal form as YYDDDF, where YY is the year and DDD is the day number. The year must be in the range 00 through 99, and the date number must be in the range 1 through 366. Note that the year is treated as 19YY. To specify a year in the range 2000-2155, you must use EXPDTX instead of EXPDT. F allows the date to remain a positive integer when converted from packed decimal to hexadecimal. All fields are right-justified.
- EXPDTX=*extended expir-date addr* specifies the address of a 4-byte field that contains the address of the expiration date of the data set. The date is given in packed decimal form as CCYYDDDF, where CC is 00 for years in the range 1900-1999, 01 for years in the range 2000-2099, and 02 for years in the range 2100-2155. The year must be in the range 00 through 99. The day must be in the range 1 through 366. The combined CCYY value cannot specify a year greater than 2155, so 0255 is the maximum value that can be specified for the combined CCYY field. All fields are right-justified. F allows the date to remain a positive integer when converted from packed decimal to hexadecimal.

  Note: Specifying 99365 or 99366 for YYDDD on EXPDT, or 0099365 or 0099366 for CCYYDDD on EXPDTX indicates an expiration date of NEVER EXPIRES. This means that an actual expiration date of 12/31/1999 cannot be specified using either of these keywords. Use the RETPD keyword with the appropriate value if the expiration date is desired.
- RETPD=*retn-period addr* specifies the address of a 2-byte binary field containing the number of days after which RACF protection for the data set expires. The value specified must be in the range 1 through 65533. To indicate that there is no expiration date, specify 65534.

If you do not specify any of these parameters, a default RACF security retention period is obtained from the RETPD keyword specified on an earlier RACF SETROPTS command.

These parameters are valid if CLASS=DATASET and DSTYPE=T.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

,FILESEQ=*number*
,FILESEQ=*reg*
specifies the file sequence number of a tape data set on a tape volume or within a tape-volume set. The *number* must be in the range 1 through 9999. If a register is specified, it must contain the file sequence number in the low-order halfword. If CLASS=DATASET and DSTYPE=T are not specified, FILESEQ is ignored.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

,GENERIC=YES
,GENERIC=ASIS
specifies whether the resource name is treated as a generic profile name. If GENERIC is specified with CLASS=DEFINE, NEWNAME, or NEWNAMX, GENERIC applies to both the old and new names. GENERIC is ignored if the GENCMD option on the RACF SETROPTS command is not specified for the class. See *z/VM: RACF Security Server Command Language Reference*.

This keyword is designed primarily for use by RACF commands.

YES
The resource name is considered a generic profile name, even if it does not contain a generic character: an asterisk (*), a percent sign (%), or, for general resource classes, an ampersand sign (&).

ASIS
The resource name is considered a generic if it contains a generic character: an asterisk (*), a percent sign (%), or, for general resource classes, an ampersand sign (&).

,INSTLN=*parm list addr*
specifies the address of an area that is to contain parameter information meaningful to the RACDEF installation exit routines. This information is passed to the installation exit routines when they are given control from the RACROUTE REQUEST=DEFINE routine.

The INSTLN parameter can be used by an application program acting as a resource manager that needs to pass information to the RACROUTE REQUEST=DEFINE routine.

,LEVEL=*number*
,LEVEL=*reg*
specifies a level value for the profile. The level number must be a valid decimal number in the range 0 to 99. If a register is specified, its low-order byte must contain the binary representation of the number.

LEVEL is valid if TYPE=DEFINE is specified.

,MCLASS=*'class name'*
,MCLASS=*class name addr*
specifies the class to which the profile defined by MENTITY= or MENTX= belongs. If an address is specified, the address must point to a 1-byte length field followed by the class name. The class name should be no longer than 8 characters. The default is MCLASS=DATASET.

,MVOLSER=*volser addr*
specifies the address of the volume serial number of the volume associated with the profile in the MENTITY operand. The field pointed to by the specified address contains the volume serial number, padded to the right with blanks if necessary to make six characters.

If you specify MENTITY or MENTX and CLASS=DATASET, you must specify MVOLSER with the name of the VOLSER or with blanks.

If you specify with blanks, the discrete MENTITY or MENTX data-set profile name must be unique, meaning it has no duplicates on the database. In this case, RACF determines the correct MVOLSER.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,MENTITY=*entity addr*
,MENTX=*extended entity address*
specifies the address of the name of the discrete or generic profile that is to be used:

- MENTITY=*entity addr* specifies the address of the name of the discrete or generic profile that is to be used as a model in defining the ENTITY or ENTITYX profile. The profile can belong to any class, as specified by the MCLASS parameter, and can be either a discrete or a generic profile.

  MENTITY can be specified with TYPE=DEFINE but not with TYPE=DEFINE,NEWNAME=*new resource name addr*.

  For data sets, the name is contained in a 44-byte field pointed to by the specified address. For general-resource classes, the length of the field is determined by the RACF class-descriptor table (CDT). The name is left-justified in the field and padded with blanks.
- MENTX=*extended entity address* specifies the address of the name of the discrete or generic profile that is to be used as a model from which to define the ENTITY or ENTITYX profile. The structure consists of two 2-byte length fields, followed by the entity name.

– The first 2-byte field specifies a buffer length that can be from 0 to 255 bytes. This length field refers to the length of the buffer that contains the entity name from which you are modeling; it does not include the length of either length field.
– The second 2-byte field specifies the actual length of the entity name from which you are modeling. This length field includes the length of the actual name without any trailing blanks; it does not include the length of either length field.

These two length fields can be used in several different ways:

– If you know the length of the entity name you are using as a model, you can specify 0 in the first field and the length of the entity name in the second field. When you specify the second field, note that each byte counts. This means the entity name that you specify will be used as a model, using the specified length.
– If you choose to place the entity name in a buffer area, specify the length of the buffer in the first field. For the second field, do one of the following:
  - If you know the length of the entity name you are using as a model, specify the length in the second field. The length of the first field can be from 0 to 255, but *must* be equal to or greater than the length of the second field.
  - If you do not know the length of the entity name you are using as a model, specify 0 in the second field, and have RACF determine the number of characters in the entity name.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

Consideration:
IBM recommends that you use MENTX rather than MENTITY. With MENTITY, the entity name you pass to RACF must be in a buffer, the size of which is determined by the length in the RACF class-descriptor table. If the maximum length of a class-descriptor entity increases in the future, you must modify your program to use a larger buffer. By using MENTX, you avoid this possible problem, because you remove the CDT dependency from your program.

The profile can belong to any class, as specified by the MCLASS parameter, and can be either a discrete or generic profile. MENTX can be specified with TYPE=DEFINE, but not with TYPE=DEFINE,NEWNAME= or TYPE=DEFINE,NEWNAMX=.

,MGENER=ASIS
,MGENER=YES
specifies whether the profile name defined by MENTITY or MENTX is to be treated as a generic name.

ASIS
The profile name is considered a generic if it contains a generic character: an asterisk (*), a percent sign (%), or, for general resource classes, an ampersand sign (&).

YES
The profile name is considered a generic, even if it does not contain a generic character: an asterisk (*), a percent sign (%) or, for general resource classes, an ampersand sign (&).

MGENER is ignored if the GENCMD option on the RACF SETROPTS command is not specified for the class. See *z/VM: RACF Security Server Command Language Reference*.

,MGMTCLA=*management type addr*
specifies the address of a management class to which the resource owner must have authority. The address must point to an 8-byte field that contains a management-class name preceded by a halfword length. If you specify MGMTCLA, you must also specify TYPE=DEFINE, RESOWN, and RELEASE=1.8.1 or a later release number.

When MGMTCLA is specified, RACROUTE REQUEST=DEFINE processing invokes REQUEST=AUTH processing to verify that the RESOWNER is authorized to the management class.

,NOTIFY=*notify-id addr*
specifies the address of an 8-byte area containing the user ID of the RACF-defined user who is to be notified when an unauthorized attempt to access the resource protected by this profile is detected.

,OWNER=*owner id addr*
specifies the address of a field containing the profile owner's ID. The owner's ID must be a valid (RACF-defined) user ID or group name. The address must point to an 8-byte field containing the owner's name, left-justified and padded with blanks.

OWNER is valid if TYPE=DEFINE is specified.

,RACFIND=YES
,RACFIND=NO
specifies whether or not a discrete profile is involved in RACROUTE REQUEST=DEFINE processing.

When TYPE=DEFINE is specified, RACFIND=YES means that a discrete profile is to be created. When the request type TYPE=DELETE, DEFINE with NEWNAME or NEWNAMX, CHGVOL, or ADDVOL is specified, RACFIND=YES means that a discrete profile already exists. The bit on the VTOC is ignored.

When TYPE=DEFINE is specified, RACFIND=NO means that no discrete profile is to be created, but some authorization checking is required. For other types of action, no discrete profile should exist.

Note: Use of RACFIND=YES with TYPE=DEFINE is not a recommended programming interface unless NEWNAME, NEWNAMX, CHGVOL, or ADDVOL is also specified. Creation of discrete profiles is intended to be done either by using the RACF command processors or by using the z/OS routines that create data sets.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=DEFINE macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time.

,RESOWN=*resource owner addr*
specifies the address of a field containing the resource owner's ID. If you specify RESOWN, you must also specify TYPE=DEFINE and the current RELEASE parameter. The resource owner's ID must be a valid (RACF-defined) user ID or group name, or *NONE*. If the resource owner's ID is specified as *NONE*, RACF performs third-party authorization checking using USERID=*NONE*. The address must point to a 2-byte field followed by the resource owner's name.

,SECLABL=*addr*
specifies the address of an 8-byte, left-justified character field containing the security label.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

An installation can use SECLABEL to establish an association between a specific RACF security level (SECLEVEL) and a set of (zero or more) RACF security categories (CATEGORY).

,SECLVL=*addr*
specifies the address of a list of installation-defined security-level identifiers. Each identifier is a halfword containing a value that corresponds to an installation-defined security-level name.

The identifiers must be in the range 1 through 254. Only one identifier may be passed in the list.

The list must start with a fullword containing the number of entries in the list (currently, only 0 or 1).

,STORCLA=*storage class addr*
specifies the address of the storage class to which the resource owner must have authority. The address must point to a 2-byte field followed by the management class name. If you specify

STORCLA, you must also specify TYPE=DEFINE, RESOWN, and RELEASE=1.8.1 or a later release number.

When specified, RACROUTE REQUEST=DEFINE processing invokes REQUEST=AUTH processing to verify that the RESOWNER is authorized to the storage class.

,TAPELBL=STD
,TAPELBL=BLP
,TAPELBL=NL
specifies the type of tape labeling to be done:

STD
IBM or ANSI standard labels

BLP
Bypass label processing

NL
Unlabeled tapes.

For TAPELBL=BLP, the user must have the requested authority to the profile ICHBLP in the general-resource class FACILITY.

For TAPELBL=NL or BLP, the user is not allowed to protect volumes with volume serial numbers in the format "L*nnnnn*".

The TAPELBL parameter is passed to the RACROUTE REQUEST=DEFINE installation exits.

This parameter is primarily intended for use by data-management routines to indicate the label type from the LABEL keyword on the JCL statement.

This parameter is valid for CLASS=DATASET and DSTYPE=T or CLASS=TAPEVOL.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

,TYPE=DEFINE
,TYPE=DEFINE,NEWNAME=*new resource name addr*
,TYPE=DEFINE,NEWNAMX=*extended new resource name addr*
,TYPE=ADDVOL,OLDVOL=*old vol addr*
,TYPE=CHGVOL,OLDVOL=*old vol addr*
,TYPE=DELETE
specifies the type of action to be taken:

Note:

- If SETROPTS ADDCREATOR is in effect when a new DATASET or general resource profile is defined, the profile creator's user ID is placed on the profile access list with ALTER authority.
- If SETROPTS NOADDCREATOR is in effect when:
  - A new generic profile is defined, the profile creator's user ID is not placed on the profile's access list. If you use profile modeling when defining a generic profile, RACF copies the access list exactly. If the creator's user ID appeared in the model's access list, the same authority is copied to the new profile.
  - A new discrete DATASET or TAPEVOL profile is defined, the profile creator's user ID is placed on the profile's access list with ALTER authority. If you use profile modeling when defining one these profiles, if the creator's user ID appeared in the model's access list, the authority is created in the new profile with ALTER authority.
  - Any other new discrete profile is defined, the profiles creator's user ID is not placed on the access list. If you use profile modeling when defining one of these profiles, RACF copies the access list exactly. If the creator's user ID appeared in the model's access list, the same authority is copied to the new profile.

DEFINE
adds the profile of the resource to the RACF database and establishes the current user as the owner of the profile.

DEFINE,NEWNAME
The address points to a field containing the new name for the resource that is to be renamed. The field should be 44 bytes when class is DATASET or the maximum name length allowed for the general-resource class.

NEWNAME is valid with CLASS=DATASET, FILE, and DIRECTRY. NEWNAME is not valid with DSTYPE=T.

DEFINE,NEWNAMX
The address points to a structure that consists of two 2-byte length fields, followed by the entity name.

- The first 2-byte field specifies a buffer length that can be from 0 to 255 bytes. This length field refers to the length of the buffer that contains the entity name; it does not include the length of either length field.
- The second 2-byte field specifies the actual length of the entity name. This length field includes the length of the actual name without any trailing blanks; it does not include the length of either length field.

These two length fields can be used in several different ways:

- If you know the length of the entity name, you can specify 0 in the first field and the length of the entity name in the second field. When you specify the second field, note that each byte counts. This means that the entity name you specify will be added to the RACF database using the specified length.
- If you choose to place the entity name in a buffer area, specify the length of the buffer in the first field. For the second field, do one of the following:
  - If you know the length of the entity name, specify the length in the second field. The length of the first field can be from 0 to 255, but *must* be equal to or greater than the length of the second field.
  - If you do not know the length of the entity name, specify 0 in the second field, and have RACF determine the number of characters in the entity name.

NEWNAMX is valid with CLASS=DATASET, FILE, and DIRECTRY. NEWNAMX is not valid with DSTYPE=T.

To use this keyword, you must also specify RELEASE=1.9 or later.

Consideration:
IBM recommends that you use NEWNAMX rather than NEWNAME. With NEWNAME, the entity name you pass to RACF must be in a buffer, the size of which is determined by the length in the RACF class-descriptor table (CDT). If the maximum length of a class-descriptor entity increases in the future, you must modify your program to use a larger buffer. By using NEWNAMX, you avoid this possible problem, because you remove the CDT dependency from your program.

The following parameters are ignored if you specify NEWNAME or NEWNAMX: ACCLVL, AUDIT, CATEGORY, DATA, ERASE, EXPDT, EXPDTX, FILESEQ, INSTLN, LEVEL, MCLASS, MENTITY, MENTX, MGENER, MVOLSER, NOTIFY, OWNER, RETPD, SECLABL, SECLVL, TAPELBL, UACC, UNIT, and WARNING.

ADDVOL
Adds the new volume to the definition of the specified resource.

For the DATASET class, the OLDVOL address specifies a previous volume of a multivolume data set.

For the TAPEVOL class, the ENTITY or ENTITYX address specifies a previous volume of a tape-volume set.

This parameter applies only to discrete profiles.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

CHGVOL
Changes the volume serial number in the definition of the specified resource from the old volume serial number identified in OLDVOL to the new volume serial number identified in the VOLSER parameter.

This parameter applies only to discrete profiles. TYPE=CHGVOL is not valid with DSTYPE=T.

On z/VM, the z/VM operating system does not use RACROUTE to provide security for tape volumes; however, you may have installed on z/VM a tape-management product that does use RACROUTE.

DELETE
Removes the profile from the RACF database. (For a multivolume data set or a tape-volume set, only the specified volume is removed from the definition.)

If DSTYPE=T is specified, the data sets must be deleted in reverse of the order in which they were created. For example, if file1 has data-set1, file2 has data-set2, and file3 has data-set3, you must do the RACROUTE REQUEST=DEFINE,TYPE=DELETE,DSTYPE=T for file3, file2, and file1, in that order.

,UACC=ALTER
,UACC=CONTROL
,UACC=UPDATE
,UACC=READ
,UACC=NONE
,UACC=*reg*
specifies a universal access authority for the profile. UACC must contain a valid access authority (ALTER, CONTROL, UPDATE, READ, or NONE).

If a register is specified, the low-order byte must contain one of the following valid access authorities:

X'80' ALTER
X'40' CONTROL
X'20' UPDATE
X'10' READ
X'01' NONE

UACC is valid if TYPE=DEFINE is specified.

,UNIT=*unit addr*
specifies the address of a field containing unit information. If a unit address is specified, the unit information in the data-set profile is replaced by the unit information pointed to by this unit address. The unit address must point to a field containing a 1-byte length field (whose value can range from 4 through 8) followed by the actual unit information. If the value in the length field is 4, the unit information is assumed to contain a copy of the information in the UCBTYP field of the UCB. Otherwise the unit information is assumed to be in the generic form (for example, 3330-1).

UNIT is valid if TYPE=CHGVOL or TYPE=DEFINE is specified and is ignored for generic names.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,WARNING=YES
,WARNING=NO
If WARNING=YES is specified, the WARNING indicator is set in the profile. Access is granted to the resource and the event is logged as a warning if either the SUCCESS or FAILURES logging is requested.

This keyword is designed primarily for use by RACF commands.

WARNING is valid if TYPE=DEFINE is specified.

,MF=S
specifies the standard form of the RACROUTE REQUEST=DEFINE macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them, using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note: Some RACROUTE parameter errors result in a RACF abend. On z/VM, these abends are simulated by RACF return and reason codes. For RACROUTE REQUEST=DEFINE, RACF return code 285 corresponds to a RACF abend that is documented in *z/VM: RACF Security Server Messages and Codes*. The reason code will also reflect the abend reason code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=DEFINE has completed successfully.

RACF Return Code
Meaning

00
RACROUTE REQUEST=DEFINE has completed successfully.

Reason Code
Meaning

00
Indicates a normal completion.

08
Indicates that MODEL was specified, but the SECLABEL value has not been copied because of one of two reasons:

- SETROPTS SECLABELCONTROL is on, but issuer is not system SPECIAL, or
- SETROPTS MLSTABLE is on, but SETROPTS MLQUIET is not

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

04
Indicates RACFIND=NO was specified and no generic profile applying to the data set was found.

04
RACROUTE REQUEST=DEFINE has completed processing.

Reason Code
Meaning

00

Indicates the following:

- For TYPE=DEFINE, the resource name was previously defined.
- For TYPE=DEFINE,NEWNAME or NEWNAMX, the new resource name was previously defined.
- For TYPE=DELETE, the resource name was not previously defined.

04

Indicates for TYPE=DEFINE that the data set name was previously defined on a different volume and that the option disallowing duplicate data sets was specified at IPL.

08

The requested function failed.

RACF Return Code
Meaning

08

RACROUTE REQUEST=DEFINE has completed processing.

Reason Code
Meaning

00

Indicates the following:

- For TYPE=DEFINE, RACF has failed the check for authority to allocate a data set or create a profile with the specified name.
- For TYPE=DELETE or TYPE=DEFINE,NEWNAME or NEWNAMX, if CHKAUTH=YES is specified, RACF has failed the authorization check.
- For TYPE=ADDVOL,OLDVOL or for TYPE=CHGVOL,OLDVOL, indicates that no profile was found that contained the specified volume and entity name.

04

Indicates for TYPE=DEFINE that no profile was found to protect the data set and that the RACF protect-all option is in effect.

08

Indicates TYPE=DEFINE (or TYPE=ADDVOL,OLDVOL or TYPE=CHGVOL,OLDVOL) and DSTYPE=T were specified, and the user is not authorized to define a data set on the specified volume, or an ADDVOL was attempted to add a forty-third volume, but the maximum number of volumes that a data set can span is 42.

0C

Indicates TYPE=DEFINE and DSTYPE=T were specified, and the user is not authorized to define a data set with the specified name.

10

Indicates DSTYPE=T or CLASS=TAPEVOL was specified, and the user is not authorized to specify TAPELBL=(,BLP)

18

Indicates that the user is not authorized to issue RACROUTE REQUEST=DEFINE when the system is in the tranquil state (when SETROPTS MLQUIET option is in effect).

1C

This can occur when PROFDEF=NO is specified in the class descriptor table.

20

Indicates the data-set owner is not authorized to use the specified DFP storage class.

24

Indicates the data-set owner is not authorized to use the specified DFP management class.

28
For CLASS=FILE or CLASS=DIRECTRY, the second qualifier in ENTITY or ENTITYX resource name is not a RACF-defined user.

2C
For TAPE data set, a security label is expected but is not specified.

30
For TAPE data set, the USER SECLABEL does not dominate the TAPE's security label when TYPE=DELETE. When type is DEFINE, ADDVOL, OR CHGVOL, the USER security label is not equal to the TAPE's security label.

34
For TYPE=DEFINE, RACF has denied the authorization to allocate the data set with that name because of one of the following:

- The profile protecting it has no security label.
- The user does not dominate the security label of the profile protecting the entity.
- The data set is not protected by any profile.

For TYPE=DEFINE, NEWNAME= or NEWNAMX=, RACF has denied the authorization to rename the data set because the new data-set name is either:

- Protected by a profile with no security label.
- Protected by a profile whose security label the user does not dominate.
- Protected by no profile.

38
The request to rename a profile is denied because the SETROPTS MLS option is in effect and one of the following is true:

- The entity is not protected by a profile.
- The entity is protected by a profile with no security label.
- The entity is protected by a profile whose security label the user cannot possibly dominate.

3C
The request to rename the resource is denied because the SETROPTS MLS option is in effect and one of the following is true:

- The new name will not be protected by any profile.
- The new name is protected by a profile with no security label.
- The user can never equal the security label that will protect the new name.

40
The request to rename the resource is denied because the security label of the generic profile protecting the new data-set name does not dominate the security label of the generic profile protecting the entity name. This is equivalent to writing down, and is disallowed because the SETROPTS MLS option is in effect.

44
The request to RACDEF is denied because the profile defined will have a security label different from the generic profile covering it and that the SETROPTS MLSTABLE option is in effect. This will happen under the following circumstances:

- The request to define a DASDVOL data set is denied because the parent generic profile has a different security label.
- The request to define a TAPEVOL data set failed for one of the following reasons:
  - The data set has a parent generic with a different security label.
  - The tape volume is not defined and a generic tape profile exists with a security label different from the security label added to the discrete tape profile.

  - The tape volume is defined and the security label being added is different from the security label of the generic profile protecting it.
- The request to add a volume is denied because the new volume will have a security label different from the security label of the generic profile protecting it.
- The request to rename the profile is denied because the security label of the generic profile covering the new name is different from the security label of the entity profile.

48

The request to REQUEST=DEFINE is failed because the user is not SPECIAL, SETROPTS GENERICOWNER is in effect, and one of the following happens:

- For TYPE=DEFINE, the user cannot define the profile because of generic owner requirements with respect to the generic profile covering the entity name. This restriction does not apply to data sets.
- For TYPE=DEFINE, NEWNAME= or NEWNAMX=, the user does not meet the generic owner requirements with respect to the less specific generic profile for NEWNAME. This reason does not apply to the DATASET class.
- For TYPE=ADDVOL, the user is not allowed to add a volume profile because a generic profile exists in the class and the user does not meet the generic owner requirement.
- For TYPE=DEFINE, DSTYPE=T, and CLASS=DATASET, the request is failed because a discrete, automatic, tape profile will be defined, but the user does not meet the generic owner requirement with respect to the generic TAPEVOL profile.

4C

Indicates that the RESOWNER is a revoked user ID.

0C

For TYPE=DEFINE,NEWNAME or NEWNAMX, the old resource name was not defined; or for CLASS=DATASET, if the generation-data-group (GDG) modeling function is active, an attempt was made to rename a GDG name to a name that requires the creation of a new profile; or if generic profile checking is active, the old resource name was protected by a generic profile and there is no generic profile that will protect the new resource name. This last case refers only to an attempt to rename an existing profile, which cannot be found.

10

For TYPE=DEFINE with MENTITY or MENTX, the model resource was not defined.

64

Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=DEFINE macro; however, the list form of the macro does not have the same RELEASE parameter. Macro processing terminates.

Example 1

Operation: Invoke RACF to define a discrete profile for a non-VSAM data set residing on the volume pointed to by register 8. Register 7 points to the data-set name. All successful requests for update authority to the data set are to be audited, as well as all unsuccessful ones.

```
RACROUTE REQUEST=DEFINE,ENTITY=(R7),VOLSER=(R8),CLASS='DATASET',  X
         AUDIT=(SUCCESS(UPDATE),FAILURES),                        X
         RACFIND=YES
```

Example 2

Operation: Use the standard form of the RACROUTE REQUEST=DEFINE macro to define a discrete data-set profile for a non-VSAM DASD data set. The data set for which you are creating a profile is a non-VSAM DASD data set named DSNAME. It resides on a volume named VOLID. You want to create a discrete

profile by specifying the RACFIND keyword. In addition, you want to notify the user called USERNAME of any access attempts that have been rejected because they exceed the UACC you are allowing for READ.

```
RACROUTE REQUEST=DEFINE,ENTITY=DSNAME,VOLSER=VOLID,    X
         CLASS='DATASET',UACC=READ,                    X
         RACFIND=YES,NOTIFY=USERNAME,RELEASE=1.7
```

Example 3

Operation: Use the standard form of the macro to check the authority of a user to define a discrete data-set profile for a non-VSAM DASD data set, but do not actually build the profile. The name of the data set is DSNAME.

```
RACROUTE REQUEST=DEFINE,ENTITY=DSNAME,VOLSER=VOLID,  X
         CLASS='DATASET',RACFIND=NO
```

Example 4

Operation: Use the standard form of the macro to define a generic data-set profile named PROFNAME. As a model for the new profile, use the discrete profile named MDELPROF whose volume serial number is in MDELVOL. Notify the user named USERNAME of any access attempts that have been rejected because they exceed the UACC you are allowing for READ.

```
RACROUTE REQUEST=DEFINE,ENTITY=PROFNAME,                    X
         CLASS='DATASET',GENERIC=YES,MENTITY=MDELPROF,      X
         MVOLSER=MDELVOL,UACC=READ,                         X
         NOTIFY=USERNAME,RELEASE=1.7
```

Example 5

Operation: Use the standard form of the macro to define a tape-volume profile for a volume whose ID is VOLID. Allow a universal access level of READ.

```
RACROUTE REQUEST=DEFINE,ENTITY=VOLID,CLASS='TAPEVOL',UACC=READ
```

Example 6

Operation: Use the standard form of the macro to delete a discrete data-set profile named DSNAME located on the volume named VOLID.

```
RACROUTE REQUEST=DEFINE,TYPE=DELETE,ENTITY=DSNAME,      X
         VOLSER=VOLID,CLASS='DATASET'
```

RACROUTE REQUEST=DEFINE (List Form)

The list form of the RACROUTE REQUEST=DEFINE macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=DEFINE macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name — *name*: Symbol. Begin *name* in column 1.

␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=DEFINE	
,ACCLVL=*access value addr*	*access value addr*: A-type address
,ACCLVL=(*access value addr,parm list addr*)	*parm list addr*: A-type address
,ACEE=*acee addr*	*acee addr*: A-type address
,AUDIT=NONE	
,AUDIT=*audit value*	*audit value*: ALL, SUCCESS, or FAILURES
,AUDIT=(*audit value(access level),audit value(access level*))	**Default:** AUDIT=READ
,CHKAUTH=YES	
,CHKAUTH=NO	**Default:** CHKAUTH=NO
,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address
	Default: CLASS=DATASET
,DATA=*data addr*	*data addr*: A-type address
,DSTYPE=N	**Default:** DSTYPE=N
,DSTYPE=V	
,DSTYPE=M	
,DSTYPE=T	
,ENTITY=*profile name addr*	*profile name addr*: A-type address
,ENTITYX=*extended profile name addr*	*extended profile name addr*: A-type address
	Note: ENTITY or ENTITYX must be specified on the list, execute, or modify form of the macro.

,ENVIR=VERIFY	Specifies that only verification is to be done. **Default:** Normal RACROUTE REQUEST=DEFINE processing.
,ERASE=YES ,ERASE=NO	**Default:** ERASE=NO
,EXPDT=*expir-date addr* ,EXPDTX=*extended expir-date addr*	*expir-date addr*: A-type address *extended expir-date addr*: A-type address
,RETPD=*retn-period addr*	*retn-period addr*: A-type address
,FILESEQ=*number*	*number*: 1-9999
,GENERIC=YES ,GENERIC=ASIS	**Default:** GENERIC=ASIS
,INSTLN=*parm list addr*	*parm list addr*: A-type address
,LEVEL=*number*	**Default:** LEVEL=zero.
,MCLASS=*'class name'* ,MCLASS=*class name addr*	*class name*: 1- to 8-character name *class name addr*: A-type address **Default:** MCLASS=DATASET
,MENTITY=*entity addr* ,MENTX=*extended entity addr*	*entity addr*: A-type address *extended entity addr*: A-type address
,MGENER=ASIS ,MGENER=YES	**Default:** MGENER=ASIS
,MGMTCLA=*management type addr*	*management type addr*: A-type address **Default:** See description of parameter.
,MVOLSER=*volser addr*	*volser addr*: A-type address
,NOTIFY=*notify-id addr*	*notify-id addr*: A-type address

,OWNER=*owner id addr*	*owner id addr*: A-type address
,RACFIND=YES ,RACFIND=NO	
,RELEASE=*number*	*number*: See Standard Form **Default:** RELEASE=1.6
,RESOWN=*resource owner addr*	*resource owner addr*: A-type address
,SECLABL=*addr*	*addr*: A-type address
,SECLVL=*addr*	*addr*: A-type address
,STORCLA=*storage class addr*	*storage class addr*: A-type address
,TAPELBL=STD ,TAPELBL=BLP ,TAPELBL=NL	**Default:** TAPELBL=STD
,TYPE=DEFINE	**Default:** TYPE=DEFINE
,TYPE=DEFINE,NEWNAME =*new resource name addr*	*new resource name addr*: A-type address
,TYPE=DEFINE,NEWNAMX =*extended new resource name addr*	*extended new resource name addr*: A-type address
,TYPE=ADDVOL,OLDVOL =*old vol addr* ,TYPE=CHGVOL,OLDVOL =*old vol addr*	*old vol addr*: A-type address
,TYPE=DELETE	
,UACC=ALTER ,UACC=CONTROL ,UACC=UPDATE ,UACC=READ ,UACC=NONE	
,UNIT=*unit addr*	*unit addr*: A-type address

,VOLSER=*vol addr*	*vol addr*: A-type address **Note:** VOLSER is required (on either LIST or EXECUTE) for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,WARNING=YES ,WARNING=NO	**Note:** Warning is valid if TYPE=DEFINE is specified.
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=DEFINE macro instruction with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=DEFINE macro instruction.

RACROUTE REQUEST=DEFINE (Execute Form)

The execute form of the RACROUTE REQUEST=DEFINE macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=DEFINE macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=DEFINE	
,ACCLVL=*access value addr*	*access value addr*: Rx-type address or register (2) - (12)
,ACCLVL=(*access value addr,parm list addr*)	*parm list addr*: Rx-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,AUDIT=NONE ,AUDIT=*audit value*	*audit value*: ALL, SUCCESS, or FAILURES
,AUDIT=(*audit value (access level),audit value(access level)*)	*access level*: READ, UPDATE, CONTROL, or ALTER

,AUDIT=*reg*	*reg*: Register (2) - (12)
,CHKAUTH=YES ,CHKAUTH=NO	
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,DATA=*data addr*	*data addr*: Rx-type address or register (2) - (12)
,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	
,ENTITY=*profile name addr*	*profile name addr*: Rx-type address
,ENTITYX=*extended profile name addr*	*extended profile name addr*: Rx-type address or register (2) - (12) **Note:** ENTITY or ENTITYX must be specified on the list, execute, or modify form of the macro.
,ENVIR=VERIFY	Specifies that only verification is to be done.
,ERASE=YES ,ERASE=NO	
,EXPDT=*expir-date addr*	*expir-date addr*: Rx-type address or register (2) - (12)
,EXPDTX=*extended expir-date addr*	*extended expir-date addr*: Rx-type address or register (2) - (12)
,RETPD=*retn-period addr*	*retn-period addr*: Rx-type address or register (2) - (12)
,FILESEQ=*number*	*number*: 1-9999
,FILESEQ=*reg*	*reg*: Register (2) - (12)
,GENERIC=YES ,GENERIC=ASIS	
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,LEVEL=*number*	

,LEVEL=*reg*	*reg*: Register (2) - (12)
,MCLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,MENTITY=*entity addr* ,MENTX=*extended entity addr*	*entity addr*: Rx-type address or register (2) - (12) *extended entity addr*: Rx-type address or register (2) - (12)
,MGENER=ASIS ,MGENER=YES	
,MGMTCLA=*management type addr*	*management type addr*: Rx-type address or register (2) - (12)
,MVOLSER=*volser addr*	*volser addr*: Rx-type address or register (2) - (12)
,NOTIFY=*notify-id addr*	*notify-id addr*: Rx-type address or register (2) - (12)
OWNER=*owner id addr*	*owner id addr*: Rx-type address or register (2) - (12)
,RACFIND=YES ,RACFIND=NO	
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=(*number*,CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6
,RESOWN=*resource owner addr*	*resource owner addr*: Rx-type address or register (2) - (12)
,SECLABL=*addr*	*addr*: Rx-type address or register (2) - (12)
,SECLVL=*addr*	*addr*: Rx-type address or register (2) - (12)
,STORCLA=*storage class addr*	*storage class addr*: Rx-type address or register (2) - (12)
,TAPELBL=STD ,TAPELBL=BLP ,TAPELBL=NL	

,TYPE=DEFINE	
,TYPE=DEFINE,NEWNAME =*new resource name addr*	*new resource name addr*: Rx-type address or register (2) - (12)
,TYPE=DEFINE,NEWNAMX =*extended new resource name addr*	*extended new resource name addr*: Rx-type address or register (2) - (12)
,TYPE=ADDVOL,OLDVOL =*old vol addr*	*old vol addr*: Rx-type address or register (2) - (12)
,TYPE=CHGVOL,OLDVOL =*old vol addr*	
,TYPE=DELETE	
,UACC=ALTER	
,UACC=CONTROL	
,UACC=UPDATE	
,UACC=READ	
,UACC=NONE	
,UACC=*reg*	*reg*: Register (2) - (12)
,UNIT=*unit addr*	*unit addr*: Rx-type address or register (2) - (12)
,VOLSER=*vol addr*	*vol addr*: Rx-type address or register (2) - (12) **Note:** VOLSER is required for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,WARNING=YES	**Note:** Warning is valid if TYPE=DEFINE is specified.
,WARNING=NO	
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=DEFINE macro instruction with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=DEFINE macro using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=DEFINE macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=DEFINE macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACROUTE REQUEST=DEFINE (Modify Form)

The modify form of the RACROUTE REQUEST=DEFINE macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=DEFINE macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=DEFINE	
,ACCLVL=*access value addr*	*access value addr*: Rx-type address or register (2) - (12)
,ACCLVL=(*access value addr,parm list addr*)	*parm list addr*: Rx-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,AUDIT=NONE	
,AUDIT=*audit value*	*audit value*: ALL, SUCCESS, or FAILURES
,AUDIT=(*audit value (access level),audit value(access level)*)	*access level*: READ, UPDATE, CONTROL, or ALTER
,AUDIT=*reg*	*reg*: Register (2) - (12)
,CHKAUTH=YES ,CHKAUTH=NO	
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)

,DATA=*data addr*	*data addr*: Rx-type address or register (2) - (12)
,DSTYPE=N	
,DSTYPE=V	
,DSTYPE=M	
,DSTYPE=T	
,ENTITY=*profile name addr*	*profile name addr*: Rx-type address
,ENTITYX=*extended profile name addr*	*extended profile name addr*: Rx-type address or register (2) - (12)
	Note: ENTITY or ENTITYX must be specified on either the list, execute, or modify form of the macro.
,ENVIR=VERIFY	Specifies that only verification is to be done.
,ERASE=YES	
,ERASE=NO	
,EXPDT=*expir-date addr*	*expir-date addr*: Rx-type address or register (2) - (12)
,EXPDTX=*extended expir-date addr*	*extended expir-date addr*: Rx-type address or register (2) - (12)
,RETPD=*retn-period addr*	*retn-period addr*: Rx-type address or register (2) - (12)
,FILESEQ=*number*	*number*: 1-9999
,FILESEQ=*reg*	*reg*: Register (2) - (12)
,GENERIC=YES	
,GENERIC=ASIS	
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,LEVEL=*number*	
,LEVEL=*reg*	*reg*: Register (2) - (12)
,MCLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)

,MENTITY=*entity addr*	*entity addr*: Rx-type address or register (2) - (12)
,MENTX=*extended entity addr*	*extended entity addr*: Rx-type address or register (2) - (12)
,MGENER=ASIS	
,MGENER=YES	
,MGMTCLA=*management type addr*	*management type addr*: Rx-type address or register (2) - (12)
,MVOLSER=*volser addr*	*volser addr*: Rx-type address or register (2) - (12)
,NOTIFY=*notify-id addr*	*notify-id addr*: Rx-type address or register (2) - (12)
,OWNER=*owner id addr*	*owner id addr*: Rx-type address or register (2) - (12)
,RACFIND=YES	
,RACFIND=NO	
,RELEASE=*number*	*number*: See Standard Form
,RELEASE=(,CHECK)	**Default:** RELEASE=1.6
,RELEASE=(*number*,CHECK)	
,RESOWN=*resource owner addr*	*resource owner addr*: Rx-type address or register (2) - (12)
,SECLABL=*addr*	*addr*: Rx-type address or register (2) - (12)
,SECLVL=*addr*	*addr*: Rx-type address or register (2) - (12)
,STORCLA=*storage class addr*	*storage class addr*: Rx-type address or register (2) - (12)
,TAPELBL=STD	
,TAPELBL=BLP	
,TAPELBL=NL	
,TYPE=DEFINE	
,TYPE=DEFINE,NEWNAME =*new resource name addr*	*new resource name addr*: Rx-type address or register (2) - (12)
,TYPE=DEFINE,NEWNAMX =*extended new resource name addr*	*extended new resource name addr*: Rx-type address or register (2) - (12)

,TYPE=ADDVOL,OLDVOL =*old vol addr*	*old vol addr*: Rx-type address or register (2) - (12)
,TYPE=CHGVOL,OLDVOL =*old vol addr*	
,TYPE=DELETE	
,UACC=ALTER	
,UACC=CONTROL	
,UACC=UPDATE	
,UACC=READ	
,UACC=NONE	
,UACC=*reg*	*reg*: Register (2) - (12)
,UNIT=*unit addr*	*unit addr*: Rx-type address or register (2) - (12)
,VOLSER=*vol addr*	*vol addr*: Rx-type address or register (2) - (12) **Note:** VOLSER is required for CLASS=DATASET and DSTYPE not equal to M when a discrete profile name is used.
,WARNING=YES	
,WARNING=NO	**Note:** Warning is valid if TYPE=DEFINE is specified.
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=DEFINE macro instruction with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=DEFINE macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=DEFINE macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=DEFINE macro using a remote control-program parameter list.

RACROUTE REQUEST=DIRAUTH: Check RACF-Directed Authorization to a Sent Message

The RACROUTE REQUEST=DIRAUTH macro works on behalf of the message-transmission managers (that is, VTAM®, TSO/E, and Session Manager) to ensure that the receiver of a message meets security-label authorization requirements. That is, the SECLABEL of the receiver of the message must dominate (be equal to or higher than) the SECLABEL of the message. On z/VM, the RACROUTE REQUEST=DIRAUTH function is provided to be compatible with z/OS, but it has limited z/VM application.

All parameter lists generated by the RACROUTE REQUEST=DIRAUTH macro are in a format that allows assembled code to be moved above 16MB of virtual storage without being reassembled.

To use this service, you must also specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=DIRAUTH must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For more details about the ICHCONN profile, see "Authorization to Issue RACROUTE Requests" on page 12.

RACROUTE REQUEST=DIRAUTH (Standard Form)

The standard form of the RACROUTE REQUEST=DIRAUTH macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

Note:
RACROUTE REQUEST=DIRAUTH requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=DIRAUTH. The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=DIRAUTH	
,RELEASE=*number*	*number*: 1.9.2 or 1.9

	Default: RELEASE=1.6 **Note:** RACROUTE macro will not allow REQUEST=DIRAUTH to be specified unless RELEASE= is also specified with a value of 1.9 or later.
,RTOKEN=*message token addr*	*message token addr*: A-type address or register (2) - (12)
,LOG=ASIS ,LOG=NOFAIL	**Default**=ASIS
,MF=S	

The parameters are explained as follows:

,LOG=ASIS
,LOG=NOFAIL
specifies the types of access attempts to the DIRAUTH resource class that RACF is to record on the SMF data set.

ASIS
RACF records the event in the manner specified on the SETR LOGOPTIONS command for the DIRAUTH resource class.

NOFAIL
If the authorization check fails, RACF does not record the attempt.

If the authorization check succeeds, RACF records the attempt as it does in ASIS.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number.

Note: RACROUTE REQUEST=DIRAUTH supports only RELEASE=1.9 or 1.9.2.

,RTOKEN=*message token addr*
specifies the address of the token of a resource (RTOKEN). The RTOKEN data contains the user token (UTOKEN) of the creator of the resource. If the first two bytes (length and version) are equal to 0, it is the same as not specifying the RTOKEN.

,MF=S
specifies the standard form of the RACROUTE REQUEST=DIRAUTH macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=DIRAUTH has completed successfully.

RACF Return Code
Meaning

00
Receiver is authorized to view the message.

Reason Code
Meaning

00
Function completed successfully.

04
RTOKEN passed belongs to an operator or a trusted user.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

04
DIRAUTH cannot make a decision.

Reason Code
Meaning

00
DIRAUTH class not active, or ACEE does not contain TOKEN information.

04
The caller was not SRB-mode compatible on ESA.

08
The definition of the provided security label was not found.

0C
The translation of the security label to its defining security level and categories failed.

10
The SECLABEL general-resource class was either not activated by SETROPTS CLASSACT(SECLABEL) or not brought into storage by SETROPTS RACLIST(SECLABEL).

14
No defining security level exists in the SECLABEL profile.

FF
An unexpected error occurred while checking security-label authorization.

0C
Invalid parameters passed to DIRAUTH.

Reason Code
Meaning

00
The resource token (RTOKEN) was not specified.

04
Invalid resource token (RTOKEN) specified.

08
The requested function failed.

RACF Return Code
Meaning

08
Receiver is not authorized to view the message.

Reason Code
Meaning

00
The security label in the user's ACEE does not currently dominate that of the message; however, the user does possess a security label that can dominate that of the message.

04
The user's security label does not dominate that of the message, and the user does not possess a security label that ever will.

Example 1

Operation: Invoke the RACROUTE REQUEST=DIRAUTH macro on behalf of the VTAM resource manager to perform security-label authorization checking in the “receiving” user's address space to ensure that the receiver's security label dominates that of the message. Specify that RACF should audit the event as specified in the SETROPTS LOGOPTIONS value for the DIRAUTH class.

In this example, the receiver's SECLABEL can never dominate the SECLABEL found in the TOKEN specified on the RTOKEN= keyword. The return code received from the DIRAUTH service is 8 and the reason code is 4.

Note: The message cannot be received by anyone other than the person to whom it was directed.

```
RACROUTE  REQUEST=DIRAUTH,WORKA=RACWK,RTOKEN=(8),       X
          LOG=ASIS,RELEASE=1.9.2
  .
  .
RACWK     DS  CL512
```

RACROUTE REQUEST=DIRAUTH (List Form)

The list form of the RACROUTE REQUEST=DIRAUTH macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=DIRAUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
ƀ	One or more blanks must precede RACROUTE.
RACROUTE	
ƀ	One or more blanks must follow RACROUTE.

REQUEST=DIRAUTH

,RELEASE=*number* — *number*: See Standard Form

,LOG=ASIS — **Default:** LOG=ASIS

,LOG=NOFAIL

,RTOKEN=*message token addr* — *message token addr*: A-type address

,MF=L

The parameters are explained under the standard form of the RACROUTE REQUEST=DIRAUTH macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=DIRAUTH macro instruction.

RACROUTE REQUEST=DIRAUTH (Execute Form)

The execute form of the RACROUTE REQUEST=DIRAUTH macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=DIRAUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name — *name*: Symbol. Begin *name* in column 1.

␣ — One or more blanks must precede RACROUTE.

RACROUTE

␣ — One or more blanks must follow RACROUTE.

REQUEST=DIRAUTH

,RELEASE=*number* — *number*: See Standard Form

,RELEASE=(,CHECK)

,RELEASE=(*number*,CHECK)

,LOG=ASIS

,LOG=NOFAIL

,RTOKEN=*message token addr*	*message token addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=DIRAUTH macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RACROUTE REQUEST=DIRAUTH supports only RELEASE=1.9 or 1.9.2.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=DIRAUTH macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=DIRAUTH macro instruction.

RACROUTE REQUEST=DIRAUTH (Modify Form)

The modify form of the RACROUTE REQUEST=DIRAUTH macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=DIRAUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=DIRAUTH	
,RELEASE=*number*	*number*: See Standard Form

,RELEASE=(,CHECK)

,RELEASE=(*number*,CHECK)

,LOG=ASIS

,LOG=NOFAIL

,RTOKEN=*message token addr* *message token addr*: Rx-type address or register (2) - (12)

,MF=(M,*ctrl addr*) *ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=DIRAUTH macro with the following exception:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RACROUTE REQUEST=DIRAUTH supports only RELEASE=1.9 or 1.9.2.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify a earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

Compatibility between the list and execute forms of the RACROUTE REQUEST=DIRAUTH macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=DIRAUTH macro instruction.

RACROUTE REQUEST=EXTRACT: Replace or Retrieve Fields

The RACROUTE REQUEST=EXTRACT macro retrieves or replaces certain specified fields from a RACF profile or encodes certain clear-text (readable) data.

When RACF is installed, the caller of RACROUTE REQUEST=EXTRACT must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

Note:

1. Encoding, replacement, and extraction are mutually exclusive.
2. The area returned by a RACROUTE REQUEST=EXTRACT or EXTRACTN request is located below 16MB.

ONLY the following REQUEST=EXTRACT functions are general-use programming interfaces:

- Retrieving or updating fields in any other product segment in the user, group, and resource profiles
- Retrieving or updating the following installation-reserved fields:
  - USERDATA

  - USRCNT
  - USRDATA
  - USRFLG
  - USRNM
- Retrieving the current or a specified user's default group or password when the password is in legacy format (encoded with DES, masking, or an installation-defined method).

Note:

The following two functions of RACROUTE REQUEST=EXTRACT are general-use programming interfaces, but are not recommended:

- Retrieving or updating fields in the BASE segment of a user, resource, or group profile
- Retrieving or updating data from the LANGUAGE segment.

Specifically, they are not recommended for use as a programming interface by IBM program products or by customer-written applications, because they may not be supported by security products other than RACF.

The following are the recommended methods for manipulating BASE and LANGUAGE segment data.

- For reading information, customers and customer programs should use
  - Output from Database Unload (IRRDBU00), or
  - A relational database created from the IRRDBU00 output.
- For reading/updating information, customers and customer programs should use RACF commands to access data.

  If a customer program needs to manipulate database information in a manner not provided by RACF commands, the functions of RACROUTE REQUEST=EXTRACT, though not recommended, are preferred over ICHEINTY.

IBM program products should not use RACROUTE REQUEST=EXTRACT to retrieve or update fields by name when those fields are in the BASE segment.

To see the names of database fields that you can retrieve and update, refer to the database template listings in Appendix B, “RACF Database Templates,” on page 293; these listings show the valid segment and field names, and the basic information content of the fields. It shows also what is and what is not part of the product interface.

RACROUTE REQUEST=EXTRACT (Standard Form)

The standard form of the RACROUTE REQUEST=EXTRACT macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

Note:

RACROUTE REQUEST=EXTRACT requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=EXTRACT.

The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=EXTRACT	
,TYPE=EXTRACT ,TYPE=EXTRACTN ,TYPE=REPLACE ,TYPE=ENCRYPT	
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,ENTITY=*profile name addr*	*profile name addr*: A-type address or register (2) - (12)
,ENTITYX=*extended profile name addr*	*extended profile name addr*: A-type address or register (2) - (12)
,FLDACC=YES ,FLDACC=NO	**Default:** FLDACC=NO
,GENERIC=ASIS ,GENERIC=YES	**Default:** GENERIC=ASIS
,RELEASE=*number*	*number*: 1.10, 1.9.2, 1.9, 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,VOLSER=*vol addr*	*vol addr*: A-type address or register (2) - (12)
,MF=S	

If TYPE=EXTRACT or EXTRACTN is specified:

,CLASS=*'class name'*	*class name*: 1- to 8-character name

,CLASS=*class name addr*	*class name addr*: A=type address or register (2) - (12) **Default:** CLASS='USER'
,DATEFMT=YYYYDDDF	
,DATEFMT=YYDDDF	**Default:** DATEFMT=YYDDDF
,DERIVE=YES	See explanation of keyword. **Default:** Normal processing
,FIELDS=*field addr*	*field addr*: A-type address or register (2) - (12)
,MATCHGN=YES	
,MATCHGN=NO	**Default:** MATCHGN=NO
,SEGMENT='*segment name*'	*segment name*: 1- to 8-character name
,SEGMENT=*segment name addr*	*segment name addr*: A-type address or register (2) - (12)
,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255 **Default:** See explanation of SUBPOOL keyword later in this section.

If TYPE=REPLACE is specified:

,CLASS='*class name*'	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12) **Default:** CLASS='USER'
,DATEFMT=YYYYDDDF	
,DATEFMT=YYDDDF	**Default:** DATEFMT=YYDDDF
,FIELDS=*field addr*	*field addr*: A-type address or register (2) - (12)
,SEGDATA=*segment data addr*	*segment data addr*: A-type address or register (2) - (12)
,SEGMENT='*segment name*'	*segment name*: 1- to 8-character name
,SEGMENT=*segment name addr*	*segment name addr*: A-type address or register (2) - (12)

If TYPE=ENCRYPT is specified:

,ENCRYPT=(*data addr*,DES) *data addr*: A-type address or register (2) - (12)

,ENCRYPT=(*data addr*,HASH)

,ENCRYPT=(*data addr*,INST)

,ENCRYPT=(*data addr*,STDDES)

Note: If TYPE=ENCRYPT is specified, the only other allowable parameters are ENTITY, ENTITYX, RELEASE, ENCRYPT, with ENCRYPT being required.

The parameters are explained as follows:

,ACEE=*acee addr*
specifies an alternate ACEE for RACF to use rather than the current ACEE. For example, for the USER class or for CLASS= not specified, if the ENTITY or ENTITYX parameter has not been specified, or ENTITYX has been specified with zero for the buffer length and zero for the actual entity name length, RACF refers to the ACEE during extract processing of user data.

If you want to use the ACEE parameter, you must specify RELEASE=1.8 or later.

The ACEE should have been created as the result of an earlier RACROUTE invocation (for example, REQUEST=VERIFY,ENVIR=CREATE).

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies the class the entity is in. The class name can be USER, GROUP, CONNECT, DATASET, or any general-resource class defined in the class-descriptor table.

If you specify CLASS, you must specify RELEASE=1.8 or later.

,DATEFMT=YYYYDDDF
,DATEFMT=YYDDDDF
specifies the format of the date that you want to extract or replace. If you specify DATEFMT=YYYYDDDF and TYPE=EXTRACT or EXTRACTN, RACF retrieves date fields in the format ccyydddF where cc=19 or cc=20. If TYPE=REPLACE is specified, RACF accepts dates in the format ccyydddF where cc=19 or cc=20. When accepting a date as input to place into the database, RACF validates that cc=19 or 20 and that:

- For cc=19, 70 < yy <= 99 and
- For cc=20, 00 <= yy <= 70.

If you specify DATEFMT=YYDDDF, RACF retrieves and accepts dates in the normal three byte format.

To specify the DATEFMT keyword, you must specify Release 1.10.

,DERIVE=YES
specifies that the desired field be obtained from the DFP segment of the appropriate profile. To specify DERIVE, you must also specify RELEASE=1.8.1 or later.

DERIVE requests are limited to the DFP segment of the DATASET and USER profiles. The following explains the DERIVE processing for both a DATASET and a USER request.

- DATASET

  Specifying the DERIVE=YES keyword with CLASS=DATASET and FIELDS=RESOWNER causes RACF to perform additional processing other than simply extracting the data-set resource owner from the data-set profile.

  DFP uses this retrieved information for authority checking when allocating a new data set.

To process the request, RACF first attempts to extract the RESOWNER field from the DATASET profile specified by the ENTITY or ENTITYX keyword. If the profile exists and the RESOWNER field contains data, RACF checks to see whether that data is the user ID of a user or group currently defined to RACF. If so, RACF returns that user ID along with a reason code that indicates whether the user ID is that of a user or a group.

If RACF does not find a profile that matches the DATASET name specified by the ENTITY or ENTITYX keyword, RACF attempts to locate the generic DATASET profile that protects that DATASET name.

If it finds the generic profile, and the RESOWNER field contains data, RACF checks to see whether that data is the user ID of a user or a group currently defined to RACF. If so, RACF returns that user ID along with a reason code that indicates whether the user ID is that of a user or a group.

If RACF does not find a generic profile, or the retrieved data is neither a user or group, RACF returns the high-level qualifier from the name specified on the ENTITY or ENTITYX keyword along with a reason code that indicates whether that high-level qualifier matches a defined user or group, or neither.

You specify a DERIVE request for RESOWNER as follows:

```
RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT,
ENTITY=DSNAME,
VOLSER=MYDASD,
CLASS='DATASET',
FIELDS=RESFLDS,
SEGMENT='DFP',
DERIVE=YES,
RELEASE=1.8.1
.........
DSNAME   DC CL44'USER1.DATASET'
MYDASD   DC CL6'DASD1'
RESFLDS  DC A(1)
         DC CL8'RESOWNER'
```

Note: You must specify all the keywords in the example for the DERIVE request to work.

- USER

  The purpose of specifying the DERIVE=YES keyword with CLASS=USER is to obtain the desired DFP field information (STORCLAS, MGMTCLAS, DATACLAS or DATAAPPL) from the profile of the user. If the user's profile does not contain the desired DFP fields, RACF goes to the user's default group and attempts to obtain the information for the remaining fields from the GROUP profile (the remaining fields being those that do not contain information in the USER profile.)

  You specify a DERIVE request for information from a USER profile as follows:

```
RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT,
ENTITY=USER01,
CLASS='USER',
FIELDS=STRFLDS,
SEGMENT='DFP',
DERIVE=YES,
RELEASE=1.8.1
.........
USER01   DC CL8'USER01'
STRFLDS  DC A(1)
         DC CL8'STORCLAS'
```

RACF processes the DERIVE keyword if it is specified with the DATASET or USER class. In addition, for DERIVE processing to occur, SEGMENT=DFP and RELEASE=1.8.1 or later must also be specified.

The DFP segment is only used in a z/OS environment.

,ENCRYPT=(*data addr*,DES)
,ENCRYPT=(*data addr*,HASH)
,ENCRYPT=(*data addr*,INST)
,ENCRYPT=(*data addr*,STDDES)
specifies the user-authentication key and authentication method.

Note: If SETROPTS PASSWORD(ALGORITHM(KDFAES)) is active and a password is being encrypted for subsequent input to RACROUTE REQUEST=VERIFY or RACROUTE REQUEST=VERIFYX with ENCRYPT=NO, then the password must be encoded using the DES method to be evaluated successfully. If a password is being encrypted for comparison with a password extracted using RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT, the comparison fails if the extracted password is encrypted using the KDFAES algorithm, even if the clear text is correct.

Specifying zero for the 1-byte length associated with the user-authentication key has the same effect as not specifying the keyword. Upon return to the caller, the first subparameter contains the address of an area that contains a 1-byte length followed by the product of the authentication process. Neither the address itself nor the length is changed. Also, data is one-way transposed; that is, no facility is provided to recover the data in readable form.

- ,ENCRYPT=(*data addr,DES*)

  Specifies the user-authentication key and the National Bureau of Standards Data Encryption Standard (DES) encryption method. The address points to a 1-byte length followed by from 1 to 255 bytes of text to be used as the key for encryption. The second subparameter specifies the RACFDES algorithm (RACF's variation of DES). When the DES algorithm is used, RACF uses the variable-length user-authentication key to encrypt eight bytes of clear-text data pointed to by the ENTITY or ENTITYX parameter, or by the user ID from the current ACEE (if no ENTITY or ENTITYX is specified or if ENTITYX is specified with zero for the buffer length and zero for the actual entity-name length).

  RACF uses the first eight bytes of clear-text data pointed to by the ENTITY or ENTITYX parameter, or the user ID specified in the ACEE. All other text is ignored.

- ,ENCRYPT=(*data addr,HASH*)

  Specifies the user-authentication key and the RACF hashing algorithm. The address points to a 1-byte length followed by from 1 to 255 bytes of text to be used as the user-authentication key. The second subparameter specifies the RACF hashing algorithm. When this hashing algorithm is used, the user-authentication key is masked instead of encrypted.

- ,ENCRYPT=(*data addr,INST*)

  Specifies the user-authentication key and the INST authentication method. The address points to a 1-byte length followed by from 1 to 255 bytes of text to be used as the key for authentication. The second subparameter specifies whatever scheme the installation is using (INST value). When the INST algorithm is used, RACF passes to the installation-defined algorithm the variable-length user-authentication key and the eight bytes of clear-text data pointed to by the ENTITY or ENTITYX parameter, or by the user ID from the current ACEE (if no ENTITY or ENTITYX is specified or if ENTITYX is specified with zero for the buffer length and zero for the actual entity-name length).

  If there is no installation-defined authentication method present, RACF uses the DES encryption method. RACF uses the first eight bytes of clear-text data pointed to by the ENTITY or ENTITYX parameter, or the user ID specified in the ACEE. All other text is ignored.

- ,ENCRYPT=(*data addr,STDDES*)

  Specifies the user-authentication key and the STDDES authentication method. The address points to a 1-byte length followed by eight bytes of text to be used as the key for authentication. The second subparameter specifies the NBS DES algorithm. When the STDDES algorithm is used, RACF uses the 8-byte user authentication key to encrypt eight bytes of clear-text data pointed to by the ENTITY or ENTITYX parameter, or the user ID from the current ACEE if no ENTITY or ENTITYX is specified or if ENTITYX is specified with zero for the buffer length and zero for the actual entity-name length.

  The authentication key must be eight bytes in length. Any other length for the key results in a parameter-list error. RACF uses the first eight bytes of clear-text data pointed to by the ENTITY or ENTITYX parameter, or the user ID specified in the ACEE. All other text is ignored.

,ENTITY=*profile name addr*
,ENTITYX=*extended profile name addr*
specifies the address:

- ,ENTITY=*profile name addr*

For Release 1.7 or earlier (limited to USER), specifies the address of an area eight bytes long that contains the resource name (user ID for CLASS=USER) for which profile data is to be extracted, or the user ID to be used when encoding. The name must be left-justified in the field and padded with blanks. If this parameter is not specified, a default value of zero indicates that RACF should use the user ID provided in the ACEE operand. If CLASS=USER is coded, information from the ACEE control block is returned in the result area.

For Release 1.8 and later, specifies the address of a resource name for which profile data is to be extracted or replaced for TYPE=EXTRACT, or REPLACE, or the clear-text data to be processed for TYPE=ENCRYPT. The area is 8 bytes long for USER and GROUP; 17 bytes long for CLASS=CONNECT; and 44 bytes long for DATASET. The lengths of all other profile names are determined by the class-descriptor table. The name must be left-justified in the field and padded with blanks. If this parameter is not specified, a default value of zero indicates that RACF should use the user ID provided in the ACEE operand. If CLASS=USER is coded, information from the ACEE control block is returned in the result area.

- ,ENTITYX=*extended profile name addr* specifies the address of a structure that consists of two 2-byte length fields, followed by the entity name.
  - The first 2-byte field specifies a buffer length that can be from 0 to 255 bytes. This length field refers to the length of the buffer that contains the entity name; it does not include the length of either length field.
  - The second 2-byte field specifies the actual length of the entity name. This length field includes the length of the actual name without any trailing blanks; it does not include the length of either length field.

  These two length fields can be used in several different ways:

  - If you know the length of the entity name, you can specify 0 in the first field and the length of the entity name in the second field. When you specify the second field, note that each byte counts. This means the entity name that you specify must match exactly the entity name on the RACF database.
  - If you choose to place the entity name in a buffer area, specify the length of the buffer in the first field. For the second field, do one of the following:
    - If you know the length of the entity name, specify the length in the second field. The length of the first field can be from 0 to 255, but *must* be equal to or greater than the length of the second field.
    - If you do not know the length of the entity name, specify 0 in the second field, and have RACF determine the number of characters in the entity name.

  If this parameter is not specified or is specified in one of the following formats, a default value of zero indicates that RACF should use the user ID from the current ACEE. These are the only two situations in which specifying zero for the buffer length and zero for the actual entity-name length does not result in a parameter-list error.

  - Zero is specified for the buffer length and zero is specified for the actual entity-name length for TYPE=ENCRYPT.
  - Zero is specified for the buffer length and zero is specified for the actual entity-name length for TYPE=EXTRACT with USER being specified for the class or CLASS= being unspecified.

Consideration:
IBM recommends that you use ENTITYX rather than ENTITY. With ENTITY, the entity name you pass to RACF must be in a buffer, the size of which is determined by the length in the RACF class-descriptor table (CDT). If the maximum length of a class-descriptor entity increases in the future, you must modify your program to use a larger buffer. By using ENTITYX, you avoid this possible problem because you removed the CDT dependency from your program.

,FIELDS=*field addr*

specifies the address of a variable-length list. The first field is a 4-byte field that contains the number of profile field names in the list that follows.

Specifying zero for this 4-byte field has the same effect as not specifying the keyword.

Each profile field name is eight bytes long, left-justified, and padded to the right with blanks. The allowable field names for each type of profile are in the template listings in Appendix B, “RACF Database Templates,” on page 293. For an illustration of how to specify the FIELDS keyword, see the TYPE=REPLACE example.

For Release=1.7 or earlier, or if you allow the keyword to default, the following options exist:

- The only acceptable value of the count field is 1.
- The only acceptable field name is PASSWORD. Use this parameter when you want to extract the user's encoded password in addition to his or her user ID and connect group. RACF returns the encoded password in the result area at an offset from the start of the area specified by the halfword at offset 4. (See the result area under TYPE=EXTRACT.)

For Release=1.8 or later, the following options exist:

- The count field can contain numbers from 1 through 255.
- The field names can be any of the field names in the template listings.

If you specify TYPE=EXTRACT or EXTRACTN, RACF retrieves the contents of the named fields from the RACF profile indicated by the CLASS= and ENTITY= or ENTITYX= parameters, and returns the contents in the result area. (See the EXTRACT keyword for an explanation of the result area.)

Beginning with Release 1.8, you can specify TYPE=REPLACE. RACF replaces or creates the indicated fields in the profile specified on the CLASS and ENTITY or ENTITYX keywords with the data pointed to by the SEGDATA keyword.

Note:

1. Do not replace a repeat group-count field. Doing so causes unpredictable results.
2. You cannot replace an entire repeat group, a single occurrence of a repeat group, or a single existing field in a repeat group. If you attempt to do so, RACF adds the data to the existing repeat group or groups.

   The only things you can do is retrieve all occurrences of specified fields within a repeat group, or add a new occurrence of a repeat group.
3. If you add occurrences of a repeat group, RACF places those additions at the beginning of the repeat group.

The following example of TYPE=REPLACE replaces fields in the BASE segment. It shows one way to code the macro and the declarations necessary to make the macro work.

```
        RACROUTE REQUEST=EXTRACT,TYPE=REPLACE,
                 CLASS='USER',
                 ENTITY=USERID,
                 FIELDS=FLDLIST,
                 SEGDATA=SEGDLIST,
                 SEGMENT=BASE

         ..............................

         USERID   DC  CL8,'BILL'
         FLDLIST  DC  A(3)
                  DC  CL8'AUTHOR'
                  DC  CL8'DFLTGRP'
                  DC  CL8'NAME'
         SEGDLIST DC  AL4(6),CL6'DJONES'
                  DC  AL4(8),CL8'SECURITY'
                  DC  AL4(11),CL11'BILL THOMAS'
         BASE     DC  CL8'BASE'
```

When the replacement action takes place, the following occurs:

- "DJONES" is placed in the AUTHOR field in the profile.
- "SECURITY" is placed in the DFLTGRP field in the profile.
- "BILL THOMAS" is placed in the 'NAME' field in the profile.

In this example, RACROUTE REQUEST=EXTRACT retrieves the UACC from a fully qualified, generic data-set profile. RACROUTE places the information in a work area in SUBPOOL 1.

```
RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT,
         VOLSER=VOLID
         CLASS='DATASET',
         ENTITY=DSN,
         FIELDS=FLDS,
         GENERIC=YES,
         SUBPOOL=1,
         RELEASE=1.8,
         SEGMENT='TSO'
 ......

 DSN  DC CL44'SYS1.LINKLIB'
 FLDS DC A(1)
      DC CL8 'UACC'
```

,FLDACC=NO
,FLDACC=YES
specifies whether field-level access checking should be performed.

If you specify FLDACC=YES, the RACF database manager checks to see that the user running your program has the authority to extract or modify the fields specified in the RACROUTE REQUEST=EXTRACT macro.

Note:

1. For field-level access checking to occur, you must specify RELEASE=1.8 or later when you code the macro. In addition, before the program executes, the security administrator must activate the FIELD class. If you code FLDACC=YES and the field class is not active, the request is failed with a return code 8, reason code 4.
2. In addition, the security administrator must issue the RDEFINE and PERMIT commands to designate those users who have the authority to access the fields designated in the RACROUTE REQUEST=EXTRACT macro.
3. If you specify FLDACC=NO or omit the parameter, the manager ignores field-level access checking.

,GENERIC=ASIS
,GENERIC=YES
specifies whether RACF is to treat the entity name as a generic profile name.

YES
RACF considers the entity name a generic profile name, even if it does not contain any of the generic characters. Characters considered generic are:

- For data set class:
  - Asterisk (*)
  - Percent (%)
- For general resource class:
  - Asterisk (*)
  - Percent (%),
  - Ampersand (&).

ASIS
RACF considers the entity name a generic profile name if it contains:

- For data set class:
  - Asterisk (*)

  - Percent (%)
- For general resource class:
  - Asterisk (*)
  - Percent (%),
  - Ampersand (&).

Note: A profile in the RACFVARS class is not considered to be a generic profile even though it contains an ampersand sign.

If you specify GENERIC, you must specify RELEASE=1.8 or later.

,MATCHGN=YES
,MATCHGN=NO
specifies that you want to extract data from a profile that matches or covers the resource name specified on the ENTITY or ENTITYX keyword.

If you specify MATCHGN=YES, RACF extracts data from the discrete profile, if one exists; if a discrete profile does not exist, RACF extracts data from the best-matching generic profile. If a best-matching generic profile is found, that profile name is returned to the caller in the ENTITY or ENTITYX location.

Note: For MATCHGN=YES, the class must be active.

If ENTITYX is specified, the length of the best-matching profile name is also returned in the 2-byte, actual entity-name-length location. If the buffer length is less than the length of the best-matching profile, you get a return and reason code indicating that the profile was not found because the buffer length specified is too small.

If you specify MATCHGN=NO, RACF extracts data from a profile (discrete or generic) that **exactly** matches the name specified on the ENTITY or ENTITYX keyword.

To specify the MATCHGN keyword, you must specify Release=1.9 or a later release number.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE 1.10 is only supported by the RACROUTE REQUEST=EXTRACT macro. Invocations of this macro can specify RELEASE=1.10 or lower. Invocations by other RACROUTE macros with RELEASE=1.10 will result in a failure.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

,SEGDATA=*segment data addr*
specifies the address of a list of data items to be placed in the respective fields named by the FIELDS= parameter. You use the SEGDATA parameter when you specify TYPE=REPLACE. If you specify SEGDATA, you must also specify CLASS, FIELDS, and RELEASE=1.8 or a later release number. The stored data is paired in the following format:

- A 4-byte length field that contains the length of the data field that follows
- A data field of variable length.

Specifying zero for the length field will cause the field being replaced to be removed from the segment. Each length field is followed immediately by a data field, until the end of the replacement data is reached. The count field, which is pointed to by the first field in the FIELDS parameter, contains the total number of length-data pairs.

,SEGMENT=*'segment name'*
,SEGMENT=*segment name addr*
specifies the RACF profile segment that RACF is to update or from which it is to extract data. If you allow the SEGMENT parameter to default, RACF assumes that you want to extract information from the base segment.

Each segment name is eight bytes long, left justified, and padded to the right with blanks. SEGMENT is not preceded by a 4-byte length field.

If you specify SEGMENT, you must also specify the CLASS and FIELDS keywords, and RELEASE=1.8 or a later release number.

,SUBPOOL=*subpool number*
specifies the storage subpool from which the extract-function routine obtains an area needed for the extraction.

On z/VM in the CMS Environment:

If this parameter is not specified, it defaults to 0. If you specify a subpool greater than 127, RACF for z/VM substitutes subpool 0.

You must adhere to the subpools supported by the CMS/OS simulation of GETMAIN. For more information, see the *z/VM: CMS Application Development Guide for Assembler*.

On z/VM in the GCS Environment:

If this parameter is not specified, it defaults to 229. If you specify a subpool, you must adhere to the subpools supported by GCS. For more information, see the *z/VM: Group Control System*.

If you are considering using the default subpool of 229 and your GCS application uses the RACROUTE REQUEST=EXTRACT macro to obtain data from RACF, the extracted data will reside in allocated storage until it is released by GCS task termination and can not be explicitly released by the GCS subtask. For long running GCS subtasks this may result in an out of storage condition. For GCS applications such as these, consider the specification of subpool 243 which enables the GCS subtask to explicitly release the storage. Note that storage allocated in subpool 243 is not released by GCS task termination, and will need to be explicitly released.

,TYPE=ENCRYPT
,TYPE=EXTRACT
,TYPE=EXTRACTN
,TYPE=REPLACE
specifies the type of function to be performed by the extract-function routine.

ENCRYPT
Allows RACF to provide an authentication token to be used in verifying a user's identity. The ENCRYPT keyword specifies the user-authentication key to be used for authentication, and the authentication method. The first eight bytes of the area pointed to by the ENTITY or ENTITYX operand is used as the clear-text data to be processed by the INST, DES, and STDDES authentication routines. The HASH method uses the user-authentication key as clear-text data and masks the data instead of encrypting it. If ENTITY or ENTITYX is not specified, or ENTITYX is specified with zero for the buffer length and zero for the actual entity-name length, the user ID from the current ACEE is used as the clear-text data. If TYPE=ENCRYPT is specified, no work area is returned.

EXTRACT
Extract information from any field in any profile.

The profile templates in Appendix B, “RACF Database Templates,” on page 293 define the type and name of each field in each profile. If you specify EXTRACT, RACF extracts information from the profile determined by the ENTITY or ENTITYX and CLASS keywords.

Specifically, RACF extracts (from the RACF database) the fields specified in the FIELDS keyword from the segment specified by the SEGMENT keyword.

Otherwise, you can obtain the default user-class information from the current user's profile (the specified or default ACEE) if you do the following:

- Specify the USER class or do not specify the CLASS= keyword.
- Do not specify the SEGMENT and FIELDS keywords.
- Do not specify the ENTITY or ENTITYX keyword, or specify ENTITYX with zero for the buffer length and zero for the actual entity-name length.

When the default information is taken from the current user's profile (the specified or default ACEE), there is no I/O to the RACF database, and the user's ID and default connect group are extracted from the current ACEE. This also results in returning the language information as follows:

- If the user's primary and secondary languages are available, they are extracted from the current ACEE, along with a code indicating that the reported languages are defined in the user's profile.
- If the user's primary and secondary languages are not available in the user's profile, the installation default primary and secondary languages set by SETROPTS are returned, along with a code indicating that the reported languages are the installation default.

Additionally, if the user's work attributes (WORKATTR) information is available, it will also be extracted from the ACEE. For the format of the WORKATTR information returned from the ACEE, see "RXTW" on page 429

To use TYPE=EXTRACT to extract field information from a profile, you must specify RELEASE=1.8 or a later release number.

Note: If you specify TYPE=EXTRACT, do not specify ENCRYPT.

Upon return, register 1 contains the address of a result area that begins with a fullword containing the length and subpool number of the area. See "RXTW" on page 429 for the mapping of this area. It is your responsibility to issue a FREEMAIN to release the area after you are through using it. See the description of the SUBPOOL keyword.

In general, RACF returns field data in the order it was specified, with a 4-byte length field preceding each profile field. The following lists show what is returned for different types of extractions:

- For a single field, you get:
  - A 4-byte length field that contains the length of the field that follows
  - If the requested field is a variable-length field, there is no additional length byte.

```
+-----------------------------------------------+
|  4 bytes of data (length of data)  |  data    |
|-----------------------------------------------+
```

- For a combination field (representing one or more fields), you receive:
  - A 4-byte length field that contains the combined length of all the fields that follow
  - A combination field made up of 4-byte length fields followed by their respective individual data fields.

```
+-------------------------------------------------------+
|   Total length of combination field                   |
+-----------------------------------------------+-------+
|  4 bytes of data (length of data1)  |  data1  |
+-----------------------------------------------+
|  4 bytes of data (length of data2)  |  data2  |
|-----------------------------------------------+
```

- For a single field within a repeat group, you receive:
  - A 4-byte length field that contains the combined length of all the fields that follow
  - A 4-byte length field that indicates the length of the specified field in the first occurrence of the repeat group. This is followed by a 4-byte length field that indicates the length of the specified field in the second occurrence of the repeat group. This order repeats until all the occurrences of the repeat group are accounted for.

```
+-------------------------------------------------------+
                          |   Total length of all the following
fields               |
```

```
Field from first              +-----------------------------------------------
+----------+
occurence of repeat group --> |  4 bytes of data (length of data1)  |  data1  |
                              +-----------------------------------------------+
Same field from next      --> |  4 bytes of data (length of data1)  |  data1  |
occurence of repeat group     +-----------------------------------------------+
```

- For a combination field (representing one or more fields) within a repeat group, you receive:
  - A 4-byte length field that contains the combined length of all the fields that follow
  - A combination field consisting of a 4-byte length field indicating the length of the individual data field that follows it, followed by the next 4-byte length field indicating the length of the next individual data field. This order repeats until all the individual fields that make up the combination field are accounted for. The order begins again for the next occurrence of the repeat group.

```
+-------------------------------------------------------+
                              |          Total length of all occurences of
the           |
                              |          combination field in the repeat
group           |
                             +-----------------------------------------------
+----------+
Combination field from first  |  4 bytes of data (length of data1)  |  data1  |
occurence of repeat group --> +-----------------------------------------------+
                              |  4 bytes of data (length of data2)  |  data2  |
                              +-----------------------------------------------+
Combination field from next   |  4 bytes of data (length of data1)  |  data1  |
occurence of repeat group --> +-----------------------------------------------+
                              |  4 bytes of data (length of data2)  |  data2  |
                              +-----------------------------------------------+
```

- When you specify the name of a repeat-group count field, you retrieve the 4-byte length followed by the 4-byte repeat group count.

  When a field to be extracted is empty, the following results:

  - For fixed length fields, RACF returns the default as specified by the template definitions. The default for flag fields is X'00'. The default for fixed-length fields in the BASE segment of the profile is binary 1(s). The default for fixed length fields in other segments is binary zeros.
  - For variable-length fields, RACF returns a length of zero and no data.

EXTRACTN

Upon return, register 1 contains the address of a result area that begins with a fullword containing the area's subpool number and length. To see the format of the result area, see the explanation of TYPE=EXTRACT, above and “RXTW” on page 429. At offset 6 in the result area, there is a flag. If the flag has a X'80', the name returned is generic.

If you specify EXTRACTN, the macro extracts information from the profile that follows the profile determined by the ENTITY or ENTITYX and CLASS keywords. From that next profile, RACF extracts the fields specified in the FIELDS keyword from the segment specified by the SEGMENT keyword. In addition, RACF returns the name of the profile from which it extracted the data.

Note:

1. If you specify TYPE=EXTRACTN, do not specify ENCRYPT=.
2. To retrieve all profiles within a class, the database must be processed twice, once to extract all discrete profiles and once again to extract all generic profiles (see “Example 2” on page 110). In exception, the DATASET class needs to be processed only once to extract all discrete and generic profiles. (See “Example 3” on page 112.)

REPLACE

Use of the REPLACE option to update a profile requires a thorough knowledge of the interrelationships of fields within a profile, and of the potential relationships between profiles. For instance, if you use RACROUTE REQUEST=EXTRACT to update a password, you should also update the password change date. However, since you cannot update the password history,

subsequent password changes (by PASSWORD or LOGON, for example) could allow the old password to be used again.

If you specify TYPE=REPLACE, RACF takes the information in the fields specified in the FIELDS parameter and pointed to by SEGDATA, and places that information in the designated segment. (The segment is within the profile determined by the ENTITY or ENTITYX and CLASS keywords.) If you specify TYPE=REPLACE, you must specify FIELDS, SEGDATA=, and RELEASE=1.8 or later. If you want to replace a segment other than the base segment, you must specify the SEGMENT keyword with the segment you want. If you do not specify SEGMENT, the segment defaults to the base segment.

Note: If you specify TYPE=REPLACE, do not specify ENCRYPT=.

,VOLSER=*volser addr*
specifies the volume serial number as follows:

- For non-VSAM DASD data sets and for tape data sets, this specifies the volume serial number of the volume on which the data set resides.
- For VSAM DASD data sets and tape data sets, this specifies the volume serial number of the catalog controlling the data set.

The field pointed to by the VOLSER address contains the volume serial number. If necessary, you must pad it to the right with blanks so it contains six characters.

If you specify VOLSER, you must specify RELEASE=1.8 or later.

VOLSER is valid with CLASS=DATASET.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,MF=S
specifies the standard form of the RACROUTE REQUEST=EXTRACT macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=EXTRACT has completed successfully.

RACF Return Code
Meaning

00
The extraction, replacement, or encoding completed successfully.

For DERIVE requests:

Reason Code
Meaning

00
Some of the values are derived from the USER profile, and some may be derived from the GROUP profile.

04
High-level qualifier returned as RESOWNER, which matched a valid USER.

08
DFP data returned from an EXTRACT request from USER profile was actually from the user's default connect group.

0C
High-level qualifier returned as RESOWNER, which matched a valid GROUP.

24
RESOWNER field matched a valid USER.

28
RESOWNER field matched a valid GROUP.

For other requests:

2C
At least one, but not all, of the fields requested failed to be retrieved because of field-level access checking.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

04
An ESTAE environment could not be established, or if Register 0 contains a reason code of 1, neither EXTRACT, EXTRACTN, REPLACE, nor ENCRYPT was specified for TYPE=.

08
For TYPE=EXTRACT, TYPE=EXTRACTN, or TYPE=REPLACE, the profile could not be found, or one of the other errors shown by the reason code has occurred.

Reason Code
Meaning

00
No profile found.

04
Field-level access checking failed. The field class may not be active.

08
Segment not found.

14
For EXTRACT:

Neither the RESOWNER field nor the high-level qualifier matched a valid USER or GROUP.

18
For MATCHGN=YES with ENTITYX= specified, the buffer length specified was too small to return the matching generic profile.

0C
RACF is inactive.

10
The extract operation failed. Register 0 contains the RACF-manager return code that caused termination. This return code is not used for the encrypt function. The manager return code and reason codes are returned in the low-order and high-order halfwords of register 0.

14
For TYPE=ENCRYPT or TYPE=EXTRACT of USER class data, ENTITY or ENTITYX was not specified and no ACEE exists or the ACEE was not for a defined user.

Reason Code
Meaning

00
No ACEE exists.

04
ACEERACF bit is off.

08
The requested function failed.

RACF Return Code
Meaning

18
A parameter-list error was encountered.

Reason Code
Meaning

04
For a TYPE=REPLACE request, FIELDS= was not specified.

08
Invalid type specified.

0C
Invalid number of fields.

10
Invalid class-name specified.

14
Invalid version in parameter list.

18
Invalid subpool specified.

1C
Invalid parameter length.

20
For TYPE=REPLACE request, SEGDATA= was not specified.

24
Invalid entity name specified.

2C
For TYPE=ENCRYPT request, no user-authentication key was specified.

30
Invalid encoding method.

34
ENTITY= or ENTITYX= was not specified with TYPE=REPLACE, TYPE=EXTRACTN, or TYPE=EXTRACT with class other than USER.

38
Multiple profiles and no volume specified.

3C
Profile found, but the wrong volume serial number was specified.

44
For the ENTITYX format, both the entity-name length and the buffer length are zero.

48
Invalid entity-name length with the ENTITY or ENTITYX keyword:

- The specified length is less than zero.
- The specified length is one of the following:
  - Greater than 44 if CLASS=DATASET
  - Greater than 8 if CLASS=USER or GROUP
  - Greater than 17 if CLASS=CONNECT
  - Greater than the maximum for the specified class as defined in the class descriptor table.
- For a TYPE=ENCRYPT request, the specified length is not zero or eight.

4C
Invalid buffer length specified with ENTITYX keyword:

- Less than zero
- Greater than 255
- Not zero but less than the entity name length.

50
The entity name contains a blank.

- If the ENTITYX keyword is specified and the entity-name length is given, the name has a blank in the beginning, in the middle, or at the end.

54
For a TYPE=ENCRYPT request for the STDDES authentication method, the specified data length is not 8.

5C
For a TYPE=EXTRACT request of user-class data that is defaulted from the ACEE, FIELDS= and SEGMENT= are not permitted because the user ID in the ACEE is not that of a RACF-defined user.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=EXTRACT macro; however, the list form of the macro does not have the same RELEASE parameter. It also indicates that the TYPE parameters specified on the list and execute forms may not be the same TYPE. Macro processing terminates.

Example 1

Operation: The following is an example of a RACROUTE REQUEST=EXTRACT that uses the STDDES authentication method to process the data in RANDATA, using the data in SESSNKEY as the authentication key. The function overlays the data in SESSNKEY with the product of the authentication process.

```
 RACROUTE REQUEST=EXTRACT,TYPE=ENCRYPT,                                X
          BRANCH=YES,                                                  X
          ENTITY=RANDATA,                                              X
          ENCRYPT=(SESSNKEY,STDDES),                                   X
          RELEASE=1.9,                                                 X
          WORKA=RACWK
..........
RANDATA DC CL8'RANDATA1'
SESSNKEY DC AL1(8),CL8'SESSNKEY'
RACWK DC CL512
```

Example 2

Operation: The following is an example of a RACROUTE REQUEST=EXTRACT with EXTRACTN. It retrieves all profiles within any class (except the DATASET class). The database must be processed twice, once to extract all discrete profiles and once to extract all generic profiles.

```
EXTRTNGR CSECT
*
*        Entry Linkage
*
         STM   14,12,12(13)             Push caller registers
         BALR  12,0                     Establish ...
         USING *,12                     ... addressability
         GETMAIN R,LV=DYNLEN            Get dynamic storage
         LR    11,1                     Move getmained address to R11
         USING DYNAREA,11               Addressability to DSECT
         ST    13,SAVEAREA+4            Save caller save area address
         LA    15,SAVEAREA              Get address of own save area
         ST    15,8(13)                 Store in caller save area
         LR    13,15                    Get address of own save area
*
*        Initialize variables in dynamic storage area
*
*
         MVC   ENTXBLEN,H6              Set buffer length to 6
         MVC   ENTXNLEN,H0              Set entity length to 0
         MVC   ENTXNAME,BLNKNAME        Set entity name to blanks
*
*        Copy static RACROUTE to dynamic GETMAINed areas
*
         MVC   DYNRACR(RACRLEN),STATRACR
*
*        Loop to retrieve the OWNER field from all discrete
*        profiles in the TAPEVOL class.
*
DISLOOP  EQU   *                        Start of discrete loop
         RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=ENTXBUFF,      *
               FIELDS=FIELDS,WORKA=WORKAREA,RELEASE=1.9,MF=(E,DYNRACR)
         LTR   15,15                    Check return code
         BNZ   TRYGEN                   Exit on nonzero return code
*                                       to search for generic profiles
*               .
*               .
*        Do discrete TAPEVOL profile processing here.
*               .
*               .
*                                       Free storage for this profile
         XR    3,3                      Zero out register 3
         XR    2,2                      Zero out register 2
         USING EXTWKEA,1                Base the result area on
*                                       register 1
         ICM   3,1,EXTWSP               Move the result area subpool
*                                       into register 3
         ICM   2,7,EXTWLN               Move the result area length
*                                       into register 2
         DROP  1                        Drop basing on register 1
         FREEMAIN R,LV=(2),A=(1),SP=(3) Free storage before processing
*                                       next profile
*
         B     DISLOOP                  Process next discrete profile
*
*
TRYGEN   EQU   *                        Search for generic profiles
*
         MVC   ENTXBLEN,H6              Set buffer length to 6
         MVC   ENTXNLEN,H0              Set entity length to 0
         MVC   ENTXNAME,BLNKNAME        Set entity name to blanks
         SLR   15,15                    Clear return code
*
*        Modify request to set GENERIC to YES
*
         RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,GENERIC=YES,           *
               RELEASE=1.9,MF=(M,DYNRACR)
*
*        Loop to retrieve the OWNER field from all generic
*        profiles in the TAPEVOL class.
*
GENLOOP  EQU   *                        Start of generic loop
*
         RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=ENTXBUFF,      *
```

```
             FIELDS=FIELDS,WORKA=WORKAREA,RELEASE=1.9,MF=(E,DYNRACR)
         LTR   15,15                    Check return code
         BNZ   DONE                     Exit on nonzero return code
*               .
*               .
*        Do generic TAPEVOL profile processing here.
*               .
*               .
*                                       Free storage for this profile
         XR    3,3                      Zero out register 3
         XR    2,2                      Zero out register 2
         USING EXTWKEA,1                Base the result area on
*                                       register 1
         ICM   3,1,EXTWSP               Move the result area subpool
*                                       into register 3
         ICM   2,7,EXTWLN               Move the result area length
*                                       into register 2
         DROP  1                        Drop basing on register 1
         FREEMAIN R,LV=(2),A=(1),SP=(3)  Free storage before processing
*                                       next profile
*
         B     GENLOOP                  Process next generic profile
*
*        Return to caller
*
DONE     EQU   *                        Return to caller
         L     13,SAVEAREA+4            Caller's save area address
         FREEMAIN R,LV=DYNLEN,A=(11)    Get dynamic storage
         LM    14,12,12(13)             Pop registers
         SLR   15,15                    Clear return code
         BR    14                       Return to caller
*
*        Static RACROUTE area
*
STATRACR RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=*-*,           *
               FIELDS=*-*,SEGMENT='BASE',CLASS='TAPEVOL',              *
               GENERIC=ASIS,WORKA=*-*,RELEASE=1.9,MF=L
RACRLEN  EQU   *-STATRACR              Length of RACROUTE
*
*        Constants
*
H0       DC    H'0'
H6       DC    H'6'
BLNKNAME DC    CL6' '
*
FIELDS   DC A(1)
         DC CL8'OWNER'
*
*        Result area mapping
*
         IRRPRXTW
*
*        Dynamic area
*
DYNAREA  DSECT
*
DYNRACR  RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=*-*,           *
               FIELDS=*-*,SEGMENT='BASE',CLASS='TAPEVOL',              *
               GENERIC=ASIS,WORKA=*-*,RELEASE=1.9,MF=L
*
*        ENTITYX structure
*
ENTXBUFF DS    0CL10                   ENTITYX structure
ENTXBLEN DS    H                       Entity name buffer length
ENTXNLEN DS    H                       Entity name actual length
ENTXNAME DS    CL6                     Entity name
*
*        Work and save areas
*
WORKAREA DS    128F                    Work area
SAVEAREA DC    18F'0'                  Save area
*
DYNLEN   EQU   *-DYNAREA                Dynamic area length
*
         END
```

Example 3

Operation: The following is an example of a RACROUTE REQUEST=EXTRACT with EXTRACTN. It retrieves all profiles within the DATASET class. The database needs to be processed only once to extract all discrete and generic profiles in the DATASET class.

```
EXTRTNDS CSECT
*
*        Entry Linkage
*
         STM   14,12,12(13)             Push caller registers
         BALR  12,0                     Establish ...
         USING *,12                     ... addressability
         GETMAIN R,LV=DYNLEN            Get dynamic storage
         LR    11,1                     Move getmained address to R11
         USING DYNAREA,11               Addressability to DSECT
         ST    13,SAVEAREA+4            Save caller save area address
         LA    15,SAVEAREA              Get address of own save area
         ST    15,8(13)                 Store in caller save area
         LR    13,15                    Get address of own save area
*
*        Initialize variables in dynamic storage area
*
*
         MVC   ENTXBLEN,H44             Set buffer length to 44
         MVC   ENTXNLEN,H0              Set entity length to 0
         MVC   ENTXNAME,BLNKNAME        Set entity name to blanks
*
*        Copy static RACROUTE to dynamic GETMAINed areas
*
         MVC   DYNRACR(RACRLEN),STATRACR
*
*        Loop to retrieve the OWNER field from all DATASET
*        profiles for each high level qualifier.  Generic
*        profiles are retrieved first.
*
LOOP     EQU   *                        Start of loop
         RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=ENTXBUFF,      *
               FIELDS=FIELDS,WORKA=WORKAREA,RELEASE=1.9,MF=(E,DYNRACR)
         LTR   15,15                    Check return code
         BNZ   DONE                     Exit on nonzero return code
*               .
*               .
*        Do DATASET profile processing here.
*               .
*               .
*                                       Free storage for this profile
         USING EXTWKEA,1                Base the result area on
*                                       register 1
         MVC   DYNGENRC,EXTFLAG         Make a local copy of generic
*                                       flag
         XR    3,3                      Zero out register 3
         ICM   3,1,EXTWSP               Move the result area subpool
*                                       into register 3
         XR    2,2                      Zero out register 2
         ICM   2,7,EXTWLN               Move the result area length
*                                       into register 2
         DROP  1                        Drop basing on register 1
         FREEMAIN R,LV=(2),A=(1),SP=(3) Free storage before processing
*                                       next profile
*
         TM    DYNGENRC,X'80'           Check generic bit
         BO    GENERIC                  Branch if generic bit is on
*                                       Otherwise, profile is not
*                                       generic, so set GENERIC to
*                                       ASIS
         RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,GENERIC=ASIS,          *
               RELEASE=1.9,MF=(M,DYNRACR)
         B     LOOP                     Process next profile
*
GENERIC  EQU   *                        Profile name is generic,
*                                       so set GENERIC to YES
         RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,GENERIC=YES,           *
               RELEASE=1.9,MF=(M,DYNRACR)
*
         B     LOOP                     Process next profile
*
*        Return to caller
*
DONE     EQU   *                        Return to caller
```

```
         L     13,SAVEAREA+4             Caller's save area address
         FREEMAIN R,LV=DYNLEN,A=(11)     Get dynamic storage
         LM    14,12,12(13)              Pop registers
         SLR   15,15                     Clear return code
         BR    14                        Return to caller
*
*        Static RACROUTE area
*
STATRACR RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=*-*,           *
               FIELDS=*-*,SEGMENT='BASE',CLASS='DATASET',              *
               GENERIC=ASIS,WORKA=*-*,RELEASE=1.9,MF=L
RACRLEN  EQU   *-STATRACR              Length of RACROUTE
*
*        Constants
*
H0       DC    H'0'
H44      DC    H'44'
BLNKNAME DC    CL44' '
*
FIELDS   DC A(1)
         DC CL8'OWNER'
*
*        Result area mapping
*
         IRRPRXTW
*
*        Dynamic area
*
DYNAREA  DSECT
*
DYNRACR  RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN,ENTITYX=*-*,           *
               FIELDS=*-*,SEGMENT='BASE',CLASS='DATASET',              *
               GENERIC=ASIS,WORKA=*-*,RELEASE=1.9,MF=L
*
*        ENTITYX structure
*
ENTXBUFF DS    0CL48                    ENTITYX structure
ENTXBLEN DS    H                        Entity name buffer length
ENTXNLEN DS    H                        Entity name actual length
ENTXNAME DS    CL44                     Entity name
*
*        Work and save areas
*
WORKAREA DS    128F                     Work area
SAVEAREA DC    18F'0'                   Save area
DYNGENRC DS    CL1                      Local copy of generic flag
*
DYNLEN   EQU   *-DYNAREA                Dynamic area length
*
         END
```

RACROUTE REQUEST=EXTRACT (List Form)

The list form of the RACROUTE REQUEST=EXTRACT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=EXTRACT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name

name: Symbol. Begin *name* in column 1.

␣

One or more blanks must precede RACROUTE.

RACROUTE

␣

One or more blanks must follow RACROUTE.

REQUEST=EXTRACT

,TYPE=EXTRACT
,TYPE=EXTRACTN
,TYPE=REPLACE
,TYPE=ENCRYPT

,ACEE=*acee addr*	*acee addr*: A-type address
,ENTITY=*profile name addr*	*profile name addr*: A-type address
,ENTITYX=*extended profile name addr*	*extended profile name addr*: A-type address
,FLDACC=YES	
,FLDACC=NO	**Default:** FLDACC=NO
,GENERIC=ASIS	
,GENERIC=YES	**Default:** GENERIC=ASIS
,RELEASE=*number*	*number*: See Standard Form **Default:** RELEASE=1.6
,VOLSER=*vol addr*	*vol addr*: A-type address

,MF=L

If TYPE=EXTRACT or EXTRACTN is specified:

,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address **Default:** CLASS='USER'
,DATEFMT=YYYYDDDF	
,DATEFMT=YYDDDF	**Default:** DATEFMT=YYDDDF
,DERIVE=YES	See explanation of keyword. **Default:** Normal processing

,FIELDS=*field addr*	*field addr*: A-type address
,SEGMENT=*'segment name'*	*segment name*: 1- to 8-character name
,SEGMENT=*segment name addr*	*segment name addr*: A-type address
,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255 **Default:** See explanation for SUBPOOL keyword in "RACROUTE REQUEST=EXTRACT (Standard Form)" on page 93.

If TYPE=REPLACE is specified:

,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address
	Default: CLASS='USER'
,DATEFMT=YYYYDDDF	
	Default: DATEFMT=YYDDDF
,FIELDS=*field addr*	*field addr*: A-type address
,MATCHGN=YES	
,MATCHGN=NO	**Default:** MATCHGN=NO
	segment data addr: A-type address
,SEGMENT=*'segment name'*	*segment name*: 1- to 8-character name
,SEGMENT=*segment name addr*	*segment name addr*: A-type address

If TYPE=ENCRYPT is specified:

,ENCRYPT=(*data addr*,DES)	*data addr*: A-type address
,ENCRYPT=(*data addr*,HASH)	
,ENCRYPT=(*data addr*,INST)	
,ENCRYPT=(*data addr*,STDDES)	

The parameters are explained under the standard form of the RACROUTE REQUEST=EXTRACT macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=EXTRACT macro.

RACROUTE REQUEST=EXTRACT (Execute Form)

The execute form of the RACROUTE REQUEST=EXTRACT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=EXTRACT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=EXTRACT	
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,ENTITY=*profile name addr*	*profile name addr*: Rx-type address or register (2) - (12)
,ENTITYX=*extended profile name addr*	*extended profile name addr*: Rx-type address or register (2) - (12)
,FLDACC=YES ,FLDACC=NO	
,GENERIC=ASIS ,GENERIC=YES	
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=(*number*,CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6
,TYPE=EXTRACT ,TYPE=EXTRACTN ,TYPE=REPLACE ,TYPE=ENCRYPT	
,VOLSER=*vol addr*	*vol addr*: Rx-type address or register (2) - (12)

,MF=(E,*ctrl addr*) *ctrl addr*: Rx-type address register (1), or register (2) - (12)

If TYPE=EXTRACT or EXTRACTN is specified:

,CLASS=*class name addr* *class name addr*: Rx-type address or register (2) - (12)

,DATEFMT=YYYYDDDF
,DATEFMT=YYDDDF **Default:** DATEFMT=YYDDDF

,DERIVE=YES See explanation of keyword.

,FIELDS=*field addr* *field addr*: Rx-type address or register (2) - (12)

,MATCHGN=YES
,MATCHGN=NO

,SEGMENT=*segment name addr* *segment name addr*: Rx-type address or register (2) - (12)

,SUBPOOL=*subpool number* *subpool number*: Decimal digit 0-255

If TYPE=REPLACE is specified:

,CLASS=*class name addr* *class name addr*: Rx-type address or Register (2) - (12)

,DATEFMT=YYYYDDDF
,DATEFMT=YYDDDF **Default:** DATEFMT=YYDDDF

,FIELDS=*field addr* *field addr*: Rx-type address or register (2) - (12)

,SEGDATA=*segment data addr* *segment data addr*: Rx-type address or register (2) - (12)

,SEGMENT=*segment name addr* *segment name addr*: Rx-type address or register (2) - (12)

If TYPE=ENCRYPT is specified:

,ENCRYPT=(*data addr*,DES) *data addr*: Rx-type address or register (2) - (12)
,ENCRYPT=(*data addr*,HASH)

,ENCRYPT=(*data addr*,INST)

,ENCRYPT=(*data addr*,STDDES)

The parameters are explained under the standard form of the RACROUTE REQUEST=EXTRACT macro with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=EXTRACT macro, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is only supported by the RACROUTE REQUEST=EXTRACT macro Invocations of this macro can specify RELEASE=1.10 or lower. Invocations by other RACROUTE macros with RELEASE=1.10 will result in failure.

Certain parameters can be specified only with particular releases. If you specify a parameter with an incompatible release level, the parameter is not accepted by macro processing. An error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=EXTRACT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACROUTE REQUEST=EXTRACT (Modify Form)

The modify form of the RACROUTE REQUEST=EXTRACT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=EXTRACT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=EXTRACT	

,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,ENTITY=*profile name addr*	*profile name addr*: Rx-type address or register (2) - (12)
,ENTITYX=*extended profile name addr*	*extended profile name addr*: Rx-type address or register (2) - (12)
,FLDACC=YES	
,FLDACC=NO	
,GENERIC=ASIS	
,GENERIC=YES	
,RELEASE=*number*	*number*: See Standard Form
,RELEASE=(,CHECK)	**Default:** RELEASE=1.6
,RELEASE=(*number*,CHECK)	
,TYPE=EXTRACT	
,TYPE=EXTRACTN	
,TYPE=REPLACE	
,TYPE=ENCRYPT	
,VOLSER=*vol addr*	*vol addr*: Rx-type address or register (2) - (12)
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address register (1), or register (2) - (12)

If TYPE=EXTRACT or EXTRACTN is specified:

,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,DATEFMT=YYYYDDDF	
,DATEFMT=YYDDDF	**Default:** DATEFMT=YYDDDF
,DERIVE=YES	See explanation of keyword.
,FIELDS=*field addr*	*field addr*: Rx-type address or register (2) - (12)
,MATCHGN=YES	
,MATCHGN=NO	

,SEGMENT=*segment name addr*	*segment name addr*: Rx-type address or register (2) - (12)
,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255

If TYPE=REPLACE is specified:

,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,DATEFMT=YYYYDDDF ,DATEFMT=YYDDDF	**Default:** DATEFMT=YYDDDF
,FIELDS=*field addr*	*field addr*: Rx-type address or register (2) - (12)
,SEGDATA=*segment data addr*	*segment data addr*: Rx-type address or register (2) - (12)
,SEGMENT=*segment name addr*	*segment name addr*: Rx-type address or register (2) - (12)

If TYPE=ENCRYPT is specified:

,ENCRYPT=(*data addr*,DES) ,ENCRYPT=(*data addr*,HASH) ,ENCRYPT=(*data addr*,INST) ,ENCRYPT=(*data addr*,STDDES)	*data addr*: Rx-type address or register (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=EXTRACT macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE 1.10 is only supported by the RACROUTE REQUEST=EXTRACT macro. Invocations of this macro can specify RELEASE=1.10 or lower. Invocations by other RACROUTE macros with RELEASE=1.10 will result in failure.

Certain parameters can be specified only with particular releases. If you specify a parameter with an incompatible release level, the parameter is not accepted by macro processing. An error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=EXTRACT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=EXTRACT macro, using a remote, control-program parameter list.

RACROUTE REQUEST=FASTAUTH: Verify Access to Resources

The RACROUTE REQUEST=FASTAUTH macro is used to check a user's authorization for access to a resource. RACROUTE REQUEST=FASTAUTH verifies access to those resources whose RACF profiles have been brought into main storage by the RACROUTE REQUEST=LIST facility.

RACROUTE REQUEST=FASTAUTH does not perform logging, gather statistics, or issue SVCs. Therefore, use of RACROUTE REQUEST=FASTAUTH is recommended only for applications that have stringent performance requirements.

Two installation exits associated with RACROUTE REQUEST=FASTAUTH can be used to make additional security checks or to instruct RACROUTE REQUEST=FASTAUTH to either accept or fail the request. You may write an application that uses RACROUTE REQUEST=LIST and RACROUTE REQUEST=FASTAUTH for authorization checking on a resource class and associated resource group that your installation defines.

When RACF is installed, the caller of RACROUTE REQUEST=FASTAUTH must have at least READ authority to the ICHCONN profile in the FACILITY class. For more details about the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

RACROUTE REQUEST=FASTAUTH (Standard Form)

The standard form of the RACROUTE REQUEST=FASTAUTH macro instruction is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

Note:
RACROUTE REQUEST=FASTAUTH requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=FASTAUTH. The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=FASTAUTH	

,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,ENTITY=*entity addr*	*entity addr*: A-type address or register (2) - (12)
,WKAREA=*area addr*	*area addr*: A-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address or register (2) - (12)
,ATTR=READ	**Default:** ATTR=READ
,ATTR=UPDATE	
,ATTR=CONTROL	
,ATTR=ALTER	
,ATTR=*reg*	*reg*: Registers (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12)
,RELEASE=*number*	*number*: 1.9.2, 1.9, 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,MF=S	

The parameters are explained as follows:

,ACEE=*acee addr*
specifies the address of the ACEE to be used to check authorization and to locate the in-storage profiles (REQUEST=LIST output) for the specified classes.

The ACEE used should have been created by a previous RACROUTE invocation (for example, REQUEST=VERIFY, ENVIR=CREATE). If no ACEE is specified, the request fails.

,APPL=*'applname'*
,APPL=*applname addr*
specifies the name of the application requesting the authorization checking. This information is not used for the authorization-checking process but is made available to the installation exit or exits. If an address is specified, it should point to an 8-byte area containing the application name, left-justified and padded with blanks if necessary.

,ATTR=READ
,ATTR=UPDATE
,ATTR=CONTROL
,ATTR=ALTER
,ATTR=*reg*
specifies the access authority the user must have to the resource profile. The following definitions apply:

READ
RACF user or group can open the resource only to read.

UPDATE
RACF user or group can open the resource to read or write.

CONTROL
For VSAM data sets, RACF user or group has authority equivalent to the VSAM control password.

For non-VSAM data sets and other resources, RACF user or group has UPDATE authority.

ALTER
RACF user or group has total control over the resource.

If a register is specified, the register must contain one of the following codes in the low-order byte of the register:

X'02' READ
X'04' UPDATE
X'08' CONTROL
X'80' ALTER

The default is READ.

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies that RACF authorization checking is to be performed for a resource of the specified class. If an address is specified, the address must point to an 8-byte field containing the class name.

,ENTITY=*entity addr*
specifies that RACF authorization checking is to be performed for the resource whose name is pointed to by the specified address. The resource name is a 6-byte volume serial number for CLASS=DASDVOL or CLASS=TAPEVOL. The name must be left-justified and padded with blanks. The length of all other resource names is determined from the class-descriptor tables.

,INSTLN=*parm list addr*
specifies the address of an area that contains information for the RACROUTE REQUEST=FASTAUTH installation exit. This address is passed to the exit routine when the exit is given control. The INSTLN parameter is used by application or installation programs to pass information to the RACROUTE REQUEST=FASTAUTH installation exit.

The address must point to a 1-byte length field, followed by the parameter list. Note that the parameter list must not contain any address.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=FASTAUTH macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. When you specify the RELEASE keyword, checking is done at assembly time.

The default is RELEASE=1.6.

,WKAREA=*area addr*
specifies the address of a 16-word work area to be used by RACROUTE REQUEST=FASTAUTH. It must contain the following information:

Word 12
The reason code that will pass back to the RACROUTE REQUEST=FASTAUTH caller using register 0.

Word 13
The return code that RACROUTE REQUEST=FASTAUTH passes back to the caller in register 15.

Word 14
The address of the in-storage profile used to determine authorization, or zero if no profile was found.

Word 15
A value provided by a preprocessing installation exit, or zero if there was no preprocessing exit. This will be passed back to the caller in register 1.

MF=S
specifies the standard form of the RACROUTE REQUEST=FASTAUTH macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=FASTAUTH has completed successfully.

RACF Return Code
Meaning

00
The user or group is authorized to use the resource.

Reason Code
Meaning

00
The RACROUTE REQUEST=FASTAUTH return code indicates whether the caller is authorized to the resource, and that the access attempt is not within the scope of the audit or global audit specification.

04
The RACROUTE REQUEST=FASTAUTH return code indicates whether the caller is authorized to the resource, and that the access attempt is within the scope of the audit or global audit specification. The RACROUTE REQUEST=FASTAUTH caller should log the attempt by issuing a RACROUTE REQUEST=AUTH for the resource that the caller is attempting to access.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

04
The resource or class name is not defined to RACF.

08
The requested function failed.

RACF Return Code
Meaning

08
The user or group is not authorized to use the resource.

Reason Code
Meaning

00
The RACROUTE REQUEST=FASTAUTH return code indicates whether the caller is authorized to the resource, and that the access attempt is not within the scope of the audit or global audit specification.

04
The RACROUTE REQUEST=FASTAUTH return code indicates whether the caller is authorized to the resource, and that the access attempt is within the scope of the audit or global audit specification. The RACROUTE REQUEST=FASTAUTH caller should log the attempt by issuing a RACROUTE REQUEST=AUTH for the resource that the caller is attempting to access.

0C
RACF is not active.

10
A RACROUTE REQUEST=FASTAUTH installation exit error occurred.

18
Indicates the profile has a conditional access list, the port-of-entry field in the security token is blank-filled, and the port-of-entry class is active.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=FASTAUTH macro; however, the list form of the macro does not have the same RELEASE parameter. Macro processing terminates.

RACROUTE REQUEST=FASTAUTH (List Form)

The list form of the RACROUTE REQUEST=FASTAUTH macro instruction is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=FASTAUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.

REQUEST=FASTAUTH	
,ACEE=*acee addr*	*acee addr*: A-type address
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address
,ATTR=READ	**Default:** ATTR=READ
,ATTR=UPDATE	
,ATTR=CONTROL	
,ATTR=ALTER	
,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address
,ENTITY=*entity addr*	*entity addr*: A-type address
,INSTLN=*parm list addr*	*parm list addr*: A-type address
,RELEASE=*number*	*number*: See Standard Form **Default:** RELEASE=1.6
,WKAREA=*area addr*	*area addr*: A-type address
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=FASTAUTH macro instruction with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=FASTAUTH macro instruction.

RACROUTE REQUEST=FASTAUTH (Execute Form)

The execute form of the RACROUTE REQUEST=FASTAUTH macro instruction is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=FASTAUTH macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.

␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE. REQUEST=FASTAUTH.
REQUEST=FASTAUTH	
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER ,ATTR=*reg*	*reg*: Register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ENTITY=*entity addr*	*entity addr*: Rx-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=(*number*,CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6
,WKAREA=*area addr*	*area addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=FASTAUTH macro instruction with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=FASTAUTH macro instruction, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by the macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=FASTAUTH macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=FASTAUTH macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACROUTE REQUEST=LIST: Build In-Storage Profiles

RACROUTE REQUEST=LIST builds in-storage profiles for RACF-defined resources. RACROUTE REQUEST=LIST processes only those resources described by class descriptors. Profiles must be built by RACROUTE REQUEST=LIST before RACROUTE REQUEST=FASTAUTH can be used to verify a user's access to a resource. The RACROUTE REQUEST=LIST macro improves resource authorization-checking performance.

When RACF is installed, the caller of RACROUTE REQUEST=LIST must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see "Authorization to Issue RACROUTE Requests" on page 12.

RACROUTE REQUEST=LIST (Standard Form)

The standard form of the RACROUTE REQUEST=LIST macro is written as follows. For a description of additional keywords that you can code and additional parameters the are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

Note:
RACROUTE REQUEST=LIST requires an ACEE. For most applications, the system will have created an ACEE to represent the active user. However, for special cases where no ACEE exists, the invoker must create one before invoking RACROUTE REQUEST=LIST. The most common way to create an ACEE is to issue a RACROUTE REQUEST=VERIFY, specifying ENVIR=CREATE. After all RACROUTE invocations are complete, the invoker should issue RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified, to delete the ACEE previously created.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
.	

REQUEST=LIST

,CLASS=*'class name'*	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address or register (2) - (12)
,ENVIR=CREATE	**Default:** ENVIR=CREATE
,ENVIR=DELETE	
,FILTER=*filter addr*	*filter addr*: A-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12)
,LIST=*list addr*	*list addr*: A-type address or register (2) - (12)
,OWNER=YES	
,OWNER=NO	**Default:** OWNER=NO
,RELEASE=*number*	*number*: 1.9.2, 1.9, 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,MF=S	

.

The parameters are explained as follows:

,ACEE=*acee addr*
specifies the address of the ACEE. RACF uses the ACEE to anchor the list of in-storage profiles.

The ACEE should have been created as the result of a previous RACROUTE invocation (for example, REQUEST=VERIFY,ENVIR=CREATE). You are required to specify the ACEE parameter.

,APPL=*'applname'*
,APPL=*applname addr*
specifies the name of the application requesting the authorization-checking. This information is not used for the authorization-checking process but is made available to the installation exit or exits. If an

address is specified, it should point to an 8-byte area containing the application name, left-justified and padded with blanks if necessary.

CLASS=*'class name'*
CLASS=*class name addr*
specifies that RACROUTE REQUEST=LIST is to build an in-storage profile for the resources of the specified class. If an address is specified, the address must point to an 8-byte field containing the class name, left-justified and padded with blanks, if necessary. The class name must be defined by a class descriptor; if not, the class is not considered to be defined.

,ENVIR=CREATE
,ENVIR=DELETE
specifies the action to be performed by the RACROUTE REQUEST=LIST macro.

CREATE
In-storage profiles for the specified class are to be built. The RACROUTE REQUEST=LIST function issues a return code of 18 if an in-storage list currently exists for the specified class.

DELETE
The in-storage profiles for the specified class are to be freed. If class is not specified, the in-storage profiles for all classes are freed.

Note:

1. The user issuing the RACROUTE REQUEST=LIST macro has the responsibility to ensure that no multitasking that results in the issuing of a RACROUTE REQUEST=AUTH, RACROUTE REQUEST=FASTAUTH, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=LIST macro occurs at the same time as the RACROUTE REQUEST=LIST.
2. When a user has issued a RACROUTE REQUEST=LIST, ENVIR=CREATE to build in-storage profiles, it is the user's responsibility to issue a RACROUTE REQUEST=LIST, ENVIR=DELETE to delete the in-storage profiles when they are no longer needed. Failure to do so may cause unpredictable results.

,FILTER=*filter addr*
specifies the address of a generic filter string, which RACF uses to search the RACF database and select profile names for which RACROUTE REQUEST=LIST builds in-storage profiles. The filter consists of a 2-byte length field followed by the filter string. The filter-string length must not exceed the length of the profile name as it is specified in the class-descriptor table.

Generic characters have special meaning when used as part of the filter string. Even when profiles do not allow an asterisk (*) in the high-level qualifier, the FILTER operand does allow it.

- % (character in a name)

  You can use the percent sign to represent any **one character** in the profile name, including a generic character. For example, if you specify DASD%% as a filter string, it can represent profile names such as DASD01, DASD2A, and DASD%5. If you specify %%%%% as a filter string, it can represent profile names such as DASD1, DASD2, DASD%, TAPE%, MY%%%, TAPE* and %%%%*.

- * (0 through *n* characters in a qualifier)

  You can use a single asterisk to represent **zero or more characters** in a qualifier, including generic characters. For example, AB*.CD can represent profile names such as AB.CD, ABEF.CD, and ABX.CD. A single asterisk can also represent an entire qualifier. For example, ABC.* represents profile names such as ABC.D, ABC.DEF, ABC.%%%, and ABC.%/DE.

- ** (0 through *n* qualifiers in a name)

  You can use a double asterisk to represent **zero or more qualifiers** in the profile name. For example, AB.**.CD represents profile names such as AB.CD, AB.DE.EF.CD, and AB.XYZ.CD. You cannot specify other characters with ** within a qualifier. For example, you can specify USER1.**, but not USER1.A**.

Note:

1. To specify the filter function, you must also specify RELEASE=1.9 or a later release number.

2. You cannot specify FILTER with LIST on the same invocation, because the two keywords are mutually exclusive.
3. You can specify the FILTER keyword with ENVIR=CREATE. If you specify ENVIR=DELETE, RACROUTE REQUEST=LIST returns a return code of 18.

,INSTLN=*parm list addr*
specifies the address of an area that contains parameter information for the RACROUTE REQUEST=LIST installation exit. The address is passed to the installation exit when the exit is given control by the RACROUTE REQUEST=LIST routine. The INSTLN parameter can be used by an application or an installation program to pass information to the RACROUTE REQUEST=LIST installation exit.

The address must point to a 1-byte length field, followed by the parameter list. Note that the parameter list must not contain any address.

,LIST=*addr*
specifies the address of a list of resource names for which RACROUTE REQUEST=LIST is to build the in-storage profiles. The list consists of a 2-byte field containing the number of the names in the list, followed by one or more variable-length names. Each name consists of a 1-byte length field, which is the length of the name, followed by the name. A zero in the 2-byte field causes the operand to be omitted.

If LIST= and FILTER= are omitted, in-storage profiles are built for all the profiles defined to RACF in the given class as well as each member for a resource grouping associated with the specified class.

Note: This operand can be specified with ENVIR=CREATE. If ENVIR=DELETE is specified, the RACROUTE REQUEST=LIST macro issues a return code of 18.

,OWNER=YES
,OWNER=NO
specifies that the resource owner is to be placed in the profile access list with the ALTER authority. If the OWNER= operand is omitted, the default is NO.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=LIST macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. When you specify the RELEASE keyword, checking is done at assembly time.

The default is RELEASE=1.6.

,MF=S
specifies the standard form of the RACROUTE REQUEST=LIST macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=LIST completed successfully.

RACF return Code
Meaning

00
RACROUTE REQUEST=LIST function completed successfully.

Reason Code
Meaning

00
Delete request successful. Create request successful, and profiles were listed.

04
Create request successful, but no profiles were listed.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

08
The specified class is not defined to RACF.

10
RACF or the resource class, or both, are not active.

14
RACLIST installation-exit error occurred.

08
The requested function failed.

RACF Return Code
Meaning

04
Unable to perform the requested function.

Reason Code
Meaning

00
Unable to establish an ESTAE environment.

01
The function code (the third byte of the parameter list) does not represent a valid function. 01 represents the RACF manager; 02 represents the RACROUTE REQUEST=LIST macro.

0C
An error was encountered during RACROUTE REQUEST=LIST processing.

18
Parameter-list error.

Reason Code
Meaning

00
No ACEE found

04
Class already RACLISTed

08
Invalid name length in list of names

0C
LIST or FILTER specified on DELETE request

10
Invalid request type (not DEFINE or DELETE)

14
LIST and FILTER specified (they are mutually exclusive)

1C
RACF is not installed or an insufficient level of RACF is installed.

20
Invalid filter sequence

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=LIST macro; however, the list form of the macro does not have the same RELEASE parameter. Macro processing terminates.

Note: If the resource class specified by the CLASS= operand is inactive, RACROUTE REQUEST=LIST does not build the in-storage profiles and a code of 0C is returned. If the resource group class is not active, RACROUTE REQUEST=LIST builds an in-storage profile but only from the individual resource profiles; resource-group profiles are ignored.

Example 1

Operation: Use the standard form of the macro to build in-storage profiles for all the profiles in the APPCLU, and chain them off the ACEE whose address is pointed to by ACEEADDR.

```
RACROUTE REQUEST=LIST,CLASS='APPCLU',ACEE=ACEEADDR,ENVIR=CREATE
```

Example 2

Operation: Use the standard form of the macro to build in-storage profiles for all the profiles whose names are in a list named PROFLIST and in the APPCLU class. Chain them from the task ACEE or address space ACEE.

```
RACROUTE REQUEST=LIST,CLASS='APPCLU',LIST=PROFLIST,ENVIR=CREATE
  .
  .
PROFLIST  DS  0CL58
PROFNUM   DC  XL2'0004'
PROF1     DC  AL1(12),CL12 'NETA.LU1.LU2'
PROF2     DC  AL1(12),CL12 'NETB.LU1.LU2'
PROF3     DC  AL1(14),CL14 'NETONE.LUA.LUB'
PROF4     DC  AL1(14),CL14 'NETTWO.LUA.LUB'
```

Example 3

Operation: Use the standard form of the macro to delete the in-storage profiles for the APPCLU class.

```
RACROUTE REQUEST=LIST,CLASS='APPCLU',ENVIR=DELETE
```

Example 4

Operation: Use the standard form of the macro to build in storage all the profiles in the specified class that match the filter string.

```
RACROUTE REQUEST=LIST,CLASS='APPCLU',ENVIR=CREATE     X
         FILTER=FILTR,RELEASE=1.9.2
   .
   .
FILTR     DS  0CL14
FILTRL    DC  XL2'000C'
FILTRT    DC  CL12 'NET*.LU*.LU*'
```

RACROUTE REQUEST=LIST (List Form)

The list form of the RACROUTE REQUEST=LIST macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=LIST macro (on page "RACROUTE REQUEST=LIST (Standard Form)" on page 128) to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
.	
REQUEST=LIST	
,ACEE=*acee addr*	*acee addr*: A-type address
,APPL='*applname*'	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address
,CLASS='*class name*'	*class name*: 1- to 8-character name
,CLASS=*class name addr*	*class name addr*: A-type address
,ENVIR=CREATE	**Default:** ENVIR=CREATE
,ENVIR=DELETE	
,FILTER=*filter addr*	*filter addr*: A-type address

INSTLN=*parm list addr*	*parm list addr*: A-type address
,LIST=*list addr*	*list addr*: A-type address
,OWNER=YES	
,OWNER=NO	**Default:** OWNER=NO
,RELEASE=*number*	*number*: See Standard Form **Default:** RELEASE=1.6
	Default: SUBPOOL=255.
,MF=L	
.	

The parameters are explained under the standard form of the RACROUTE REQUEST=LIST macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=LIST macro instruction.

RACROUTE REQUEST=LIST (Execute Form)

Refer to the Standard Form of the RACROUTE REQUEST=LIST macro (on page "RACROUTE REQUEST=LIST (Standard Form)" on page 128) to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

The execute form of the RACROUTE REQUEST=LIST macro is written as follows.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
.	
REQUEST=LIST	
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)

,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ENVIR=CREATE ,ENVIR=DELETE	
,FILTER=*filter addr*	*filter addr*: Rx-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,LIST=*list addr*	*list addr*: Rx-type address or register (2) - (12)
,OWNER=YES ,OWNER=NO	
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=(*number*,CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6
,MF=(E,*,ctrl addr*)	*ctrl addr*: Rx-type address or register (2) - (12)

.

The parameters are explained under the standard form of the RACROUTE REQUEST=LIST macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=LIST macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify a parameter with an incompatible release level, the parameter is not accepted by macro processing.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=LIST macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=LIST macro instruction, using a remote, control-program parameter list.

RACROUTE REQUEST=LIST (Modify Form)

The modify form of the RACROUTE REQUEST=LIST macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=LIST macro (on page "RACROUTE REQUEST=LIST (Standard Form)" on page 128) to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
.	
REQUEST=LIST	
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,FILTER=*filter addr*	*filter addr*: Rx-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,ENVIR=CREATE ,ENVIR=DELETE	
,LIST=*list addr*	*list addr*: Rx-type address or register (2) - (12)
,OWNER=YES ,OWNER=NO	
,RELEASE=*number*	*number*: See Standard Form

,RELEASE=(,CHECK) **Default:** RELEASE=1.6

,RELEASE=(*number*,CHECK)

,MF=(M,*ctrl addr*) *ctrl addr*: Rx-type address or register (2) - (12)

.

The parameters are explained under the standard form of the RACROUTE REQUEST=LIST macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=LIST macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify a parameter with an incompatible release level, the parameter is not accepted by macro processing.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=LIST macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=LIST macro instruction, using a remote, control-program parameter list.

RACROUTE REQUEST=STAT: Determine RACF Status

The RACROUTE REQUEST=STAT macro determines if RACF is active and, optionally, determines whether a given resource class is defined to RACF. If a resource class name is defined to RACF, the macro also determines whether the class is active.

To use this service, you must also specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=STAT must have at least READ authority to the ICHCONN profile in the FACILITY class. For more details on the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

RACROUTE REQUEST=STAT (Standard Form)

The standard form of the RACROUTE REQUEST=STAT macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

name *name*: Symbol. Begin *name* in column 1.

␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=STAT	
,RELEASE=*number*	*number*: 1.9.2 or 1.9 **Default:**RELEASE=1.6
,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,ENTRY=*entry addr*	*entry addr*: A-type address or register (2) - (12)
,MF=S	

The parameters are explained as follows:

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies the class name for which RACF authorization checking is performed. The name can be explicitly defined on the macro by enclosing the name in quotes. If specified, the address must point to an 8-byte field containing the class name, left-justified and padded with blanks if necessary. If CLASS= is omitted, the status of RACF is returned.

The class name specified must be a general resource defined to RACF in the class descriptor table. For information on the IBM-supplied classes, see "IBM-Supplied Class Descriptor Table Entires" in *z/VM: RACF Security Server Macros and Interfaces*.

Note: The classes DATASET, USER, and GROUP are not in the class descriptor table.

,ENTRY=*entry addr*
specifies the address of a 4-byte area that is set to the address of the specified class in the RACF class-descriptor table. This operand is ignored when the CLASS= operand is omitted.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=STAT macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

Certain parameters can be specified only with particular releases. If you specify a parameter with an incompatible release level, the parameter is not accepted by the macro processing. An error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Execution-time validation of the compatibility between the list and execute forms of the RACROUTE REQUEST=STAT macro can be done by specifying the CHECK subparameter on the execute form of the macro.

The release specified must be 1.9 or higher.

,MF=S
specifies the standard form of the RACROUTE REQUEST=STAT macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=STAT has completed successfully.

RACF Return Code
Meaning

00
RACF is active and, if CLASS= was specified, the class is active.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing can take place.

04
RACF is active; the class is inactive.

08
RACF is active; the class is not defined to RACF.

0C
RACF is inactive.

10
RACF is inactive; the class is inactive.

14
RACF is inactive.

1C
Incorrect parameter-list length is detected for the request-specific portion of the RACROUTE REQUEST=STAT parameter list. The parameter list length is not decimal 12.

18
RACF is not installed, or an insufficient level of RACF is installed.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=STAT macro; however, the list form of the macro does not have the same RELEASE parameter. Macro processing terminates.

Note: The class-descriptor entry for the specified class is returned to the caller (in the 4-byte area addressed by the entry address for return codes 00, 04, 0C, and 10.

Example 1

Operation: Determine whether the DASDVOL class is active, and retrieve the address of its class descriptor. A fullword, CDADDR, contains the class-descriptor address.

```
RACROUTE REQUEST=STAT,CLASS='DASDVOL',ENTRY=CDADDR
```

RACROUTE REQUEST=STAT (List Form)

The list form of the RACROUTE REQUEST=STAT macro instruction is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=STAT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=STAT	
,RELEASE=*number*	*number*: See Standard Form
,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address
,ENTRY=*entry addr*	*entry addr*: A-type address
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=STAT macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=STAT macro.

RACROUTE REQUEST=STAT (Execute Form)

The execute form of the RACROUTE REQUEST=STAT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=STAT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=STAT	
,RELEASE=*number* ,RELEASE=(*number*,CHECK)	*number*: See Standard Form
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ENTRY=*entry addr*	*entry addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=STAT macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=STAT macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify a parameter with an incompatible release level, the parameter is not accepted by the macro processing. An error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=STAT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=STAT macro, using a remote, control-program parameter list.

RACROUTE REQUEST=STAT (Modify Form)

The modify form of the RACROUTE REQUEST=STAT macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=STAT macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=STAT	
,RELEASE=*number* ,RELEASE=(*number*,CHECK)	*number*: See Standard Form
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ENTRY=*entry addr*	*entry addr*: Rx-type address or register (2) - (12)
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=STAT macro, with the following exceptions:

,RELEASE=*number*
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=STAT macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify a parameter with an incompatible release level, the parameter is not accepted by the macro processing. An error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=STAT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=STAT macro, using a remote, control-program parameter list.

RACROUTE REQUEST=TOKENBLD: Build a UTOKEN

The RACROUTE REQUEST=TOKENBLD macro builds a UTOKEN. The TOKNIN keyword specifies the location of the existing token from which a modified token is to be built. Note that the modification does not change the input token; instead, the function builds a new, modified token from the parameters provided. The TOKNOUT keyword specifies the location where the new, modified token is to be built.

The following order of priority exists for building the UTOKEN:

- Keywords specified on the request take precedence over corresponding fields in the TOKNIN and STOKEN parameters.
- All fields within the token specified by the TOKNIN keyword take precedence over those specified by STOKEN.
- The fields for the submitter's ID, submitter's group, submit node, execution node, session, port of entry and its class, as obtained from the token specified by the STOKEN keyword, are last.

If you do not want certain fields overridden, do not specify keywords for those fields.

To use this service, you must specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=TOKENBLD must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see "Authorization to Issue RACROUTE Requests" on page 12.

RACROUTE REQUEST=TOKENBLD (Standard Form)

The standard form of the RACROUTE REQUEST=TOKENBLD macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENBLD	
,RELEASE=*number*	*number*: 1.9.2 or 1.9 **Default:**RELEASE=1.6
,TOKNOUT=*output token addr*	*output token addr*: A-type address or register (2) - (12)
,EXENODE=*execution node addr*	*execution node addr*: A-type address or register (2) - (12)

,GROUP=*group addr*	*group addr*: A-type address or register (2) - (12)
,POE=*port of entry addr*	*port of entry addr*: A-type address or register (2) - (12)
,REMOTE=YES	
,REMOTE=NO	**Default:** REMOTE=NO
,SECLABL=*seclabel addr*	*seclabel addr*: A-type address or register (2) - (12)
,SESSION=*type*	**Default:** SESSION=TSO
,SGROUP=*submitting group addr*	*submitting group addr*: A-type address or register (2) - (12)
,SNODE=*submitting node addr*	*submitting node addr*: A-type address or register (2) - (12)
,STOKEN=*stoken addr*	*stoken addr*: A-type address or register (2) - (12)
,SUSERID=*submitting userid addr*	*submitting userid addr*: A-type address or register (2) - (12)
,TERMID=*terminal addr*	*terminal addr*: A-type address or register (2) - (12)
,TOKNIN=*input token addr*	*input token addr*: A-type address or register (2) - (12)
,TRUSTED=YES	
,TRUSTED=NO	**Default:** TRUSTED=NO
,USERID=*userid addr*	*userid addr*: A-type address or register (2) - (12)
,MF=S	

The parameters are explained as follows:

,EXENODE=*execution node addr*
specifies the address of an area that contains a 1-byte length field followed by the name of the node on which the unit of work is to be executed. The node name cannot exceed eight bytes.

,GROUP=*group addr*
specifies the group of the user who has entered the system. The address points to a 1-byte length field followed by the group name. The group name cannot exceed eight bytes.

,POE=*port of entry addr*
specifies the address of the port of entry into the system. The address points to the name of the input device through which the user or job entered the system. For example, this may be the name of the terminal logged on. The port of entry is an 8-character field, left-justified and padded with blanks.

The port of entry (POE) becomes a part of the user's security token (UTOKEN). A flag in the UTOKEN uniquely identifies the RACF general-resource class to which the data in the POE field belongs.

The TERMINAL class covers the terminal used to log on to z/VM.

When both the POE and TERMID keywords are specified, the POE keyword takes precedence.

REMOTE=YES
REMOTE=NO
specifies whether the job came through the network. The default is REMOTE=NO.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=TOKENBLD macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time.

The release specified must be 1.9 or higher.

,SECLABL=*seclabel addr*
specifies the address of an 8-byte, left-justified field, which contains the security label, padded to the right with blanks.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

An installation can use security labels to establish an association between a specific RACF security level (SECLEVEL) and a set of (zero or more) RACF security categories (CATEGORY). If it is necessary to use security labels to prevent the unauthorized movement of data from one level to another when multiple levels of data are in use on the system at the same time, see *z/VM: RACF Security Server Security Administrator's Guide* for further information.

,SESSION=*type*
specifies the session type to be associated with the request. Session types are literals. When the SESSION keyword is used in combination with the POE keyword, SESSION determines the class with which the POE keyword is connected.

The default session type is TSO. This is the only valid session type. This session type refers to any interactive session, such as z/VM logon.

,SGROUP=*submitting group addr*
specifies the address of an area that contains a 1-byte length field followed by the group ID of the user who submitted the unit of work. The group ID cannot exceed eight bytes.

,SNODE=*submitting node addr*
specifies the address of an area that contains a 1-byte length field followed by the name of the node from which the unit of work was submitted. The node name cannot exceed eight bytes.

,STOKEN=*stoken addr*
specifies the address of the submitter's UTOKEN. The first byte contains the length of the UTOKEN, and the second byte contains the version number. See the ICHRUTKN mapping, “RUTKN” on page 421 for the current version and release.

If you specify STOKEN, the user ID in STOKEN becomes the submitter's ID in TOKNOUT, unless you specify the submitter's ID (SUSER) keyword. In this case, that keyword becomes the submitter's ID in TOKNOUT. Likewise, if you specified GROUP in STOKEN, that becomes the submitter's group in TOKNOUT, unless you specified the submitter's group (SGROUP) keyword. The SESSION, port of entry

(POE), and port-of-entry class (POEX) fields are also used from the STOKEN. The execution node becomes the resulting submit node and execution node, unless you specify the submit node (SNODE) or execution node (EXENODE) keywords. In all cases, the specified keywords on the request override the fields of STOKEN, if one is specified.

,SUSERID=*submitting userid addr*
specifies the address of an area that contains a 1-byte length field followed by the user ID of the user who submitted the unit of work. The user ID cannot exceed eight bytes.

,TERMID=*terminal addr*
specifies the address of the identifier of the terminal through which the user is accessing the system. The address points to an 8-byte area containing the terminal identifier. The area must reside in a non-task-related storage subpool.

,TOKNIN=*input token addr*
specifies the address of the UTOKEN or RTOKEN that is to be used as a base for the output token.

,TOKNOUT=*output token addr*
specifies the address of the caller-provided area for the modified token data. The first byte of storage at the address specified must contain the token length. The second byte must contain the format version of the token. This provides for downward compatibility with all versions of the token map.

For a description of the fields that are used from STOKEN by TOKNOUT, see the STOKEN description.

,TRUSTED=YES
,TRUSTED=NO
specifies whether or not the unit of work is a member of the trusted computer base. Subsequent RACROUTE REQUEST=AUTH requests using a token with this attribute have the following effects:

- Authorization checking is bypassed (this includes bypassing the checks for security classification on users and data)
- No statistics are updated
- No audit records are generated, except those requested using the SETROPTS LOGOPTIONS command.

,USERID=*userid addr*
specifies the identification of the operator who has entered the system. The address points to a 1-byte length field followed by the user ID. The user ID cannot exceed eight bytes.

,MF=S
specifies the standard form of the RACROUTE REQUEST=TOKENBLD macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=TOKENBLD has completed successfully.

RACF Return Code
Meaning

08
Indicates REQUEST=TOKENBLD has completed successfully.

Reason Code
Meaning

10
TOKNOUT area specified was larger than expected; on return the token-length field contains the expected length.

14
STOKEN area specified was larger than expected.

20
TOKNIN area specified was larger than expected.

08
The requested function failed.

RACF Return Code
Meaning

00
An error occurred before the function could initiate.

Reason Code
Meaning

00
A recovery environment could not be established.

Example 1

Operation: The following example shows how a RACROUTE REQUEST=TOKENBLD macro can be specified to replace a SECLABEL in an existing token.

```
    RACROUTE REQUEST=TOKENBLD,TOKNOUT=TOKOUT,TOKNIN=TOKIN        X
             SECLABL=SLBL,RELEASE=1.9
   .
   .
TOKOUT    DS  0CL80
          DC  XL2'5001' /* FIRST 2 BYTES SPECIFY TOKEN VERSION */
          DC  XL78'0'

TOKIN     DS  0CL80
          DC  XL2'5001'
          DC  XL78'0'

SLBL      DC  CL8'INTERNAL'
```

Note: Additional keywords required by RACF to complete the request, such as WORKA, are specified on RACROUTE itself.

RACROUTE REQUEST=TOKENBLD (List Form)

The list form of the RACROUTE REQUEST=TOKENBLD macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENBLD macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
ƀ	One or more blanks must precede RACROUTE.
RACROUTE	
ƀ	One or more blanks must follow RACROUTE.

REQUEST=TOKENBLD

,RELEASE=*number*	*number*: See Standard Form **Default:**RELEASE=1.6
,EXENODE=*execution node addr*	*execution node addr*: A-type address
,GROUP=*group addr*	*group addr*: A-type address
,POE=*port of entry addr*	*port of entry addr*: A-type address
,REMOTE=YES ,REMOTE=NO	**Default:** REMOTE=NO
,SECLABL=*seclabel addr*	*seclabel addr*: A-type address
,SESSION=*type*	**Default:** SESSION=TSO
,SGROUP=*submitting group addr*	*submitting group addr*: A-type address
,SNODE=*submitting node addr*	*submitting node addr*: A-type address
,STOKEN=*stoken addr*	*stoken addr*: A-type address
,SUSERID=*submitting userid addr*	*submitting userid addr*: A-type address
,TERMID=*terminal addr*	*terminal addr*: A-type address
,TOKNIN=*input token addr*	*input token addr*: A-type address
,TOKNOUT=*output token addr*	*output token addr*: A-type address
,TRUSTED=YES ,TRUSTED=NO	**Default:** TRUSTED=NO
,USERID=*userid addr*	*userid addr*: A-type address

,MF=L

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENBLD macro instruction with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=TOKENBLD macro instruction.

RACROUTE REQUEST=TOKENBLD (Execute Form)

The execute form of the RACROUTE REQUEST=TOKENBLD macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENBLD macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENBLD	
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=(*number*,CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6
,EXENODE=*execution node addr*	*execution node addr*: Rx-type address or register (2) - (12)
,GROUP=*group addr*	*group addr*: Rx-type address or register (2) - (12)
,POE=*port of entry addr*	*port of entry addr*: Rx-type address or register (2) - (12)
,REMOTE=YES ,REMOTE=NO	
,SECLABL=*seclabel addr*	*seclabel addr*: Rx-type address or register (2) - (12)
,SESSION=*type*	*type*: Any valid session type

,SGROUP=*submitting group addr*	*submitting group addr*: Rx-type address or register (2) - (12)
,SNODE=*submitting node addr*	*submitting node addr*: Rx-type address or register (2) - (12)
,STOKEN=*stoken addr*	*stoken addr*: Rx-type address or register (2) - (12)
,SUSERID=*submitting userid addr*	*submitting userid addr*: Rx-type address or register (2) - (12)
,TERMID=*terminal addr*	*terminal addr*: Rx-type address or register (2) - (12)
,TOKNIN=*input token addr*	*input token addr*: Rx-type address or register (2) - (12)
,TOKNOUT=*output token addr*	*output token addr*: Rx-type address or register (2) - (12)
,TRUSTED=YES ,TRUSTED=NO	
,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENBLD macro instruction with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=TOKENBLD macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

Compatibility between the list and execute forms of the RACROUTE REQUEST=TOKENBLD macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=TOKENBLD macro, using a remote, control-program parameter list.

RACROUTE REQUEST=TOKENBLD (Modify Form)

The modify form of the RACROUTE REQUEST=TOKENBLD macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENBLD macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
ƀ	One or more blanks must precede RACROUTE.
RACROUTE	
ƀ	One or more blanks must follow RACROUTE.
REQUEST=TOKENBLD	
,RELEASE=*number*	*number*: See Standard Form
,EXENODE=*execution node addr*	*execution node addr*: Rx-type address or register (2) - (12)
,GROUP=*group addr*	*group addr*: Rx-type address or register (2) - (12)
,POE=*port of entry addr*	*port of entry addr*: Rx-type address or register (2) - (12)
,REMOTE=YES ,REMOTE=NO	
,SECLABL=*seclabel addr*	*seclabel addr*: Rx-type address or register (2) - (12)
,SESSION=*type*	*type*: Any valid session type
,SGROUP=*submitting group addr*	*submitting group addr*: Rx-type address or register (2) - (12)
,SNODE=*submitting node addr*	*submitting node addr*: Rx-type address or register (2) - (12)
,STOKEN=*stoken addr*	*stoken addr*: Rx-type address or register (2) - (12)

,SUSERID=*submitting userid addr*	*submitting userid addr*: Rx-type address or register (2) - (12)
,TERMID=*terminal addr*	*terminal addr*: Rx-type address or register (2) - (12)
,TOKNIN=*input token addr*	*input token addr*: Rx-type address or register (2) - (12)
,TOKNOUT=*output token addr*	*output token addr*: Rx-type address or register (2) - (12)
,TRUSTED=YES ,TRUSTED=NO	
,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENBLD macro instruction with the following exceptions:

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=TOKENBLD macro, using a remote, control-program parameter list.

RACROUTE REQUEST=TOKENMAP: Access Token Fields

The RACROUTE REQUEST=TOKENMAP macro maps a token in either internal or external format. Internal format is the encoded data format returned from a RACROUTE REQUEST=VERIFYX or a RACROUTE REQUEST=TOKENXTR. External format is the user-readable format that is mapped by the ICHRUTKN macro. See “RUTKN” on page 421. RACROUTE REQUEST=TOKENMAP is the **only** interface used to map token data.

The primary purpose of the RACROUTE REQUEST=TOKENMAP function is to allow a caller to access individual fields within the UTOKEN. The caller needs to provide the proper length and version for the corresponding format version of a token.

To use this service, you must specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=TOKENMAP must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For more details on the ICHCONN profile, see “Authorization to Issue RACROUTE Requests” on page 12.

RACROUTE REQUEST=TOKENMAP (Standard Form)

The standard form of the RACROUTE REQUEST=TOKENMAP macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.

RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENMAP	
,RELEASE=*number*	*number*: 1.9.2 or 1.9 **Default:**RELEASE=1.6
,TOKNIN=*input token addr*	*input token addr*: A-type address or register (2) - (12)
,TOKNOUT=*output token addr*	*output token addr*: A-type address or register (2) - (12)
,FORMOUT=INTERNAL	
,FORMOUT=EXTERNAL	**Default:** FORMOUT=EXTERNAL
,MF=S	

The parameters are explained as follows:

,FORMOUT=EXTERNAL
,FORMOUT=INTERNAL
specifies the format of the output token area.

INTERNAL
This is the encoded data format that is returned form a RACROUTE REQUEST=VERIFYX, RACROUTE REQUEST=TOKENXTR or a RACROUTE REQUEST=TOKENBLD.

EXTERNAL
The user-readable format is that which is mapped by the ICHRUTKN macro. See “RUTKN” on page 421.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=TOKENMAP macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

,TOKNIN=*input token addr*
specifies the address of the UTOKEN or RTOKEN that is to be converted to internal or external format.

,TOKNOUT=*output token address*
specifies the address of the caller-provided area for the converted token data. The first byte of storage at the address specified must contain the token length. The second byte must contain the format version of the token. This provides for downward compatibility with all versions of the token map.

,MF=S
specifies the standard form of the RACROUTE REQUEST=TOKENMAP macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note: Some RACROUTE parameter errors result in a RACF abend. On z/VM, these abends are simulated by RACF return and reason codes. For RACROUTE REQUEST=TOKENMAP, RACF return code 9C7 corresponds to a RACF abend, which is documented in *z/VM: RACF Security Server Messages and Codes*. The reason code also reflects the abend reason code.

Note to reader:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=TOKENMAP has completed successfully.

RACF Return Code
Meaning

00
Reason described by the following hexadecimal reason codes:

Reason Code
Meaning

00
The request was successful.

04
TOKEN was not converted; already in requested format.

0C
TOKNOUT area too large; token was successfully extracted.

04
RACROUTE REQUEST=TOKENMAP did not complete successfully.

Example 1

Operation: The following is an example of invoking the TOKENMAP function:

```
RACROUTE REQUEST=TOKENMAP,                                          X
         TOKNIN=TOKIN,TOKNOUT=TOKOUT,                               X
         RELEASE=1.9,WORKA=RACWK

TOKIN     DS  0CL80
          DC  XL2'5001'  /*FIRST 2 BYTES SPECIFY TOKEN VERSION */
          DC  XL78'0'

TOKOUT    DS  0CL80
          DC  XL2'5001'
          DC  XL78'0'

RACWK     DS  CL'512'
```

RACROUTE REQUEST=TOKENMAP (List Form)

The list form of the RACROUTE REQUEST=TOKENMAP macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENMAP macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENMAP	
,RELEASE=*number*	*number*: See Standard Form
,FORMOUT=EXTERNAL	
,TOKNIN=*input token addr*	*input token addr*: A-type address
,TOKNOUT=*output token addr*	*output token addr*: A-type address
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENMAP macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=TOKENMAP macro instruction.

RACROUTE REQUEST=TOKENMAP (Execute Form)

The execute form of the RACROUTE REQUEST=TOKENMAP macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENMAP macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENMAP	

,RELEASE=*number*	*number*: See Standard Form
,FORMOUT=INTERNAL ,FORMOUT=EXTERNAL	
,TOKNIN=*input token addr*	*input token addr*: Rx-type address or register (2) - (12)
,TOKNOUT=*output token addr*	*output token addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENMAP macro with the following exception:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=TOKENMAP macro instruction.

RACROUTE REQUEST=TOKENMAP (Modify Form)

The modify form of the RACROUTE REQUEST=TOKENMAP macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENMAP macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENMAP	
,RELEASE=*number*	*number*: See Standard Form
,FORMOUT=INTERNAL ,FORMOUT=EXTERNAL	
,TOKNIN=*input token addr*	*input token addr*: Rx-type address or register (2) - (12)

,TOKNOUT=*output token addr*	*output token addr*: Rx-type address or register (2) - (12)
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE macro with the following exception:

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE macro instruction.

RACROUTE REQUEST=TOKENXTR: Extract UTOKENS

The RACROUTE REQUEST=TOKENXTR macro extracts a UTOKEN from the current address space, task or a caller-specified ACEE. Any information not available from the ACEE is returned as blanks, or is defaulted. The ICHRUTKN macro maps the UTOKEN. See "RUTKN" on page 421.

To use this service, you must specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=TOKENXTR must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For more details on the ICHCONN profile, see "Authorization to Issue RACROUTE Requests" on page 12.

RACROUTE REQUEST=TOKENXTR (Standard Form)

The standard form of the RACROUTE REQUEST=TOKENXTR macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENXTR	
,RELEASE=*number*	*number*: 1.9.2 or 1.9
,TOKNOUT=*output token addr*	*output token addr*: A-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,MF=S	

The parameters are explained as follows:

,ACEE=*acee addr*
specifies the address of the ACEE from which information is to be extracted.

The ACEE should have been created as the result of a previous RACROUTE invocation (for example, REQUEST=VERIFY,ENVIR=CREATE).

Note: If no ACEE is specified or found, a token is returned with the following information:

- The user ID is '*'
- A default TOKEN flag is on
- An undefined user flag is on
- A flag indicating this token is created for a pre-RACF 1.9 system.

If there is a down-level ACEE, a token is returned with indication of a pre-RACF 1.9 system; certain fields may be blank or zero, such as SECLABEL, submitting user ID, and other information not available for a down-level ACEE.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=TOKENXTR macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number.

When you specify the RELEASE keyword, checking is done at assembly time.

The release specified must be 1.9 or higher.

,TOKNOUT=*return token addr*
specifies the address where the requester wants TOKENXTR to return the UTOKEN that was extracted from the ACEE. The first byte of storage at the address specified must contain the number of bytes of available storage. The second byte must contain the format version of the token.

,MF=S
specifies the standard form of the RACROUTE REQUEST=TOKENXTR macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note: Some RACROUTE parameter errors result in a RACF abend. On z/VM, these abends are simulated by RACF return and reason codes. For RACROUTE REQUEST=TOKENXTR, RACF return code 9C7 corresponds to a RACF abend, which is documented in *z/VM: RACF Security Server Messages and Codes*. The reason code also reflects the abend reason code.

Note to reader:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=TOKENXTR has completed successfully.

RACF Return Code
Meaning

00
Reason described by the following hex reason codes:

Reason Code
Meaning

00
The request was successful.

04
Invalid (down level) ACEE supplied. Information is defaulted if it could not be extracted.

08
No ACEE available. Information is defaulted if it could not be extracted.

0C
TOKNOUT area length was too large.

04
RACROUTE REQUEST=TOKENXTR did not complete successfully.

Example 1

Operation: This example shows how to extract information from the ACEE to determine which type of label to put on printed output data. Please refer to the example section for TOKENMAP for an example of converting the token returned by TOKENXTR into readable form.

```
RACROUTE  REQUEST=TOKENXTR,TOKNOUT=TOKOUT,                          X
          WORKA=RACWK,                                              X
          RELEASE=1.9
  .
  .
RACWK     DS  CL512
TOKOUT    DS  0CL80
          DC  XL2'5001' /*FIRST 2 BYTES SPECIFY THE TOKEN VERSION */
          DC  XL78'0'
```

RACROUTE REQUEST=TOKENXTR (List Form)

The list form of the RACROUTE REQUEST=TOKENXTR macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENXTR macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENXTR	
,RELEASE=*number*	*number*: See Standard Form
,ACEE=*acee addr*	*acee addr*: A-type address

,TOKNOUT=*output token addr* — *output token addr*: A-type address

,MF=L

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENXTR macro with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=TOKENXTR macro instruction.

RACROUTE REQUEST=TOKENXTR (Execute Form)

The execute form of the RACROUTE REQUEST=TOKENXTR macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENXTR macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name — *name*: Symbol. Begin *name* in column 1.

␣ — One or more blanks must precede RACROUTE.

RACROUTE

␣ — One or more blanks must follow RACROUTE.

REQUEST=TOKENXTR

,RELEASE=*number* — *number*: See Standard Form

,ACEE=*acee addr* — *acee addr*: Rx-type address or register (2) - (12)

,TOKNOUT=*output token addr* — *output token addr*: Rx-type address or register (2) - (12)

,MF=(E, *ctrl addr*) — *ctrl addr*: Rx-type address or register (1) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENXTR macro with the following exception:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=TOKENXTR macro instruction.

RACROUTE REQUEST=TOKENXTR (Modify Form)

The modify form of the RACROUTE REQUEST=TOKENXTR macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=TOKENXTR macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=TOKENXTR	
,RELEASE=*number*	*number*: See Standard Form
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,TOKNOUT=*output token addr*	*output token addr*: Rx-type address or register (2) - (12)
,MF=M	

The parameters are explained under the standard form of the RACROUTE REQUEST=TOKENXTR macro with the following exception:

,MF=M
specifies the modify form of the RACROUTE REQUEST=TOKENXTR macro instruction.

RACROUTE REQUEST=VERIFY: Identify and Verify a RACF-Defined User

The RACROUTE REQUEST=VERIFY macro provides RACF user identification and verification. The macro instruction identifies a user and verifies that the user is defined to RACF and has supplied at least one of the following:

- a valid password
- a valid password phrase
- a valid MFA credential

You can protect applications by using profiles in the APPL class along with this macro to control the users able to use applications. For more information on protecting applications, see *z/VM: RACF Security Server Security Administrator's Guide*.

The following order of priority exists for replacing the fields in the existing TOKEN:

- Keywords specified on the request take precedence over corresponding fields in the TOKNIN and STOKEN parameters.
- All fields within the token specified by the TOKNIN keyword take precedence over those specified by STOKEN.
- The fields for the submitter's ID, submitter's group, submit node, execution node, session, port of entry and its class, as obtained from the token specified by the STOKEN keyword are last.

If you do not want certain fields overridden, do not specify keywords for those fields.

When RACF is installed, the caller of RACROUTE REQUEST=VERIFY must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see "Authorization to Issue RACROUTE Requests" on page 12.

RACROUTE REQUEST=VERIFY (Standard Form)

The standard form of the RACROUTE REQUEST=VERIFY macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=VERIFY	
,ACEE=*address of fullword*	*address of fullword*: A-type address or register (2) - (12)
,ACTINFO=*account addr*	*account addr*: A-type address or register (2) - (12)
,APPL='*applname*' ,APPL=*applname addr*	*applname*: 1- to 8-character name *applname addr*: A-type address or register (2) - (12)
,ENCRYPT=YES ,ENCRYPT=NO	**Default:** ENCRYPT=YES
,ENVIR=CREATE ,ENVIR=VERIFY ,ENVIR=CHANGE ,ENVIR=DELETE	**Default:** ENVIR=CREATE

,EXENODE=*execution node addr*
: *execution node addr*: A-type address or register (2) - (12)

,GROUP=*group addr*
: *group addr*: A-type address or register (2) - (12)

,INSTLN=*parm list addr*
: *parm list addr*: A-type address or register (2) - (12)

,JOBNAME=*jobname addr*
: *jobname addr*: A-type address or register (2) - (12)

,LOG=ASIS
,LOG=ALL
,LOG=NONE
: **Default:** LOG=ASIS

,LOGSTR=*logstr addr*
: *logstr addr*: A-type address or register (2) - (12)

,NEWPASS=*new password addr*
: *new password addr*: A-type address or register (2) - (12)

,NEWPHRASE=*new password phrase addr*
: *new password phrase addr*: A-type address or register (2) - (12)

,PASSCHK=YES
,PASSCHK=NO
,PASSCHK=NOMFA
: **Default:** PASSCHK=YES

,PASSWRD=*password addr*
: *password addr*: A-type address or register (2) - (12)

,PGMNAME=*programmer name addr*
: *programmer name addr*: A-type address or register (2) - (12)

,PHRASE=*password phrase addr*
: *password phrase addr*: A-type address or register (2) - (12)

,POE=*port of entry addr*
: *port of entry addr*: A-type address or register (2) - (12)

,RELEASE=*number*
: *number*: 1.9.2, 1.9, 1.8.1, 1.8, 1.7, or 1.6
Default: RELEASE=1.6

,REMOTE=YES
,REMOTE=NO
: **Default:** REMOTE=NO

,SECLABL=*seclabel addr*	*seclabel addr*: A-type address or register (2) - (12)
,SESSION=*type*	**Default:** SESSION=TSO
,SGROUP=*submitting group addr*	*submitting group addr*: A-type address or register (2) - (12)
,SNODE=*submitting node addr*	*submitting node addr*: A-type address or register (2) - (12)
,STAT=ASIS ,STAT=NO	**Default:** STAT=ASIS
,STOKEN=*stoken addr*	*stoken addr*: A-type address or register (2) - (12)
,SUBPOOL=*subpool number*	**Default:** See explanations for the SUBPOOL keyword later in this section.
,SUSERID=*submitting userid addr*	*submitting userid addr*: A-type address or register (2) - (12)
,TERMID=*terminal addr*	*terminal addr*: A-type address or register (2) - (12)
,TOKNIN=*utoken addr*	*utoken addr*: A-type address or register (2) - (12)
,TOKNOUT=*utoken addr*	*utoken addr*: A-type address or register (2) - (12)
,TRUSTED=YES ,TRUSTED=NO	**Default:** TRUSTED=NO
,USERID=*userid addr*	*userid addr*: A-type address or register (2) - (12)
,MF=S	

The parameters are explained as follows:

,ACEE=*address of fullword*
specifies the address of a fullword to be used as described below.

For ENVIR=DELETE

specifies the address of a fullword that contains the address of the ACEE to be deleted.

For ENVIR=CHANGE

specifies the address of a fullword that contains the address of the ACEE to be changed.

For ENVIR=CREATE

specifies the address of a fullword in which the RACROUTE REQUEST=VERIFY function places the address of the ACEE created.

The ACEE is created in the RACF service machine, so the addresses in that ACEE point to data in the RACF service machine. This can result in confusion when the ACEE is returned to the invoker; therefore, if a copy of this ACEE is returned to the invoker, all addresses are replaced with zeros. Fields in the ACEE that are not addresses contain normal data.

When the invoker uses this ACEE on another invocation, (for example, REQUEST=VERIFY,ENVIR=CHANGE), the RACF service machine restores the missing addresses from a copy of the ACEE that it keeps in the machine.

If an ACEE is not specified, a copy of the ACEE is created in the RACF service machine. The ACEE is only kept for the duration of the request. When the request is complete, the ACEE is automatically deleted.

Note: If you omit USERID, GROUP, and PASSWRD and if you code ENVIR=CREATE or if ENVIR=CREATE is used as the default, you receive a return code of X'00' and obtain an ACEE that contains an asterisk (*) (X'5C') in place of the USERID and group name.

,ACTINFO=*account addr*
specifies the address of a field containing accounting information. This 144-byte area is passed to the RACINIT installation exit routine; it is not used by the RACROUTE REQUEST=VERIFY routine. The accounting field, if supplied, should have the following format:

- The first byte of the field 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 data.

,APPL=*'applname'*
,APPL=*applname addr*
specifies the name of the application issuing the RACROUTE REQUEST=VERIFY to verify the user's authority to access the application. This saves the application from having to do a separate RACHECK.

If an address is specified, the address must point to an 8-byte application name, left-justified and padded with blanks if necessary.

,ENCRYPT=YES
,ENCRYPT=NO
specifies whether RACROUTE REQUEST=VERIFY encodes the old password and the new password passed to it.

The default is YES.

YES
Signifies that the data specified by the PASSWRD and NEWPASS keywords are not preencoded. RACROUTE REQUEST=VERIFY encodes the data before storing it in the user profile or using it to compare against stored data.

NO
Signifies that the data specified by the PASSWRD and NEWPASS keywords are already encoded. RACROUTE REQUEST=VERIFY bypasses the encoding of this data before storing it in or comparing it against the user profile.

Notes:

1. ENCRYPT=NO does not apply to PHRASE and NEWPHRASE and will be ignored if specified.

2. If a RACF password in the RACF database is encrypted using KDFAES, then the data specified by the PASSWRD keyword must be encoded using the DES method in order to be evaluated successfully. If the KDFAES algorithm is active, then the data specified by the NEWPASS keyword must be encoded using the DES method in order to create a new password that will be evaluated correctly.

,ENVIR=CREATE
,ENVIR=VERIFY
,ENVIR=CHANGE
,ENVIR=DELETE

specifies the action to be performed by the user initialization component regarding the ACEE.

The default is CREATE.

CREATE
The user should be verified and an ACEE created.

VERIFY
Only a user verification is to be made; however, it can optionally be combined with updating the user's password. The installation can do this through an SAF installation exit. If the installation does not use SAF to satisfy this request, the RACROUTE caller receives a return code of 4, with RACF return and reason codes of zero. The request is not processed by the RACF SVC.

CHANGE
The ACEE should be modified according to other parameters specified on RACROUTE REQUEST=VERIFY. You can change only the connect group with this option.

DELETE
The ACEE should be deleted. This parameter should be used only if a previous RACROUTE REQUEST=VERIFY has completed successfully.

Both copies of the ACEE, the copy in the user's storage and the copy in the RACF service machine's storage, are deleted.

Attention: IBM recommends issuing a RACROUTE REQUEST=VERIFY,ENVIR=DELETE to delete only an ACEE that you created. See "Special Considerations for Changing or Deleting an ACEE" on page 173 for other options.

ENVIR=CHANGE and ENVIR=DELETE may not be specified with the parameters as identified in the following table.

Restricted Parameters	ENVIR=CHANGE	ENVIR=DELETE
APPL=	X	X
EXENODE=	X	X
GROUP=		X
NEWPASS=	X	X
NEWPHRASE=	X	X
PASSWRD=	X	X
PHRASE=	X	X
POE=	X	X
REMOTE=	X	X
SECLABL=	X	X
SESSION=	X	X
SGROUP=	X	X
SNODE=	X	X

Restricted Parameters	ENVIR=CHANGE	ENVIR=DELETE
STOKEN=	X	X
SUSERID=	X	X
TERMID=	X	X
TOKNIN=	X	X
TRUSTED=	X	X
USERID=	X	X

,EXENODE=*execution node addr*
specifies the address of an area that contains a 1-byte length field followed by the name of the node on which the unit of work is to be executed. The node name cannot exceed eight bytes.

,GROUP=*group addr*
specifies the group specified by the user who has entered the system. The address points to a 1-byte length field, followed by the group name, which can be up to eight characters.

,INSTLN=*parm list addr*
specifies the address of an area containing parameter information meaningful to the RACINIT installation exit routine. This area is passed to the installation exit when the exit routine is given control from the RACROUTE REQUEST=VERIFY routine.

The INSTLN parameter can be used by an installation having a user-verification or job-initiation application, and wanting to pass information from one installation module to the RACINIT installation exit routine.

The address must point to a 1-byte length field, followed by the parameter list. Note that the parameter list must not contain any address.

,JOBNAME=*jobname addr*
specifies the address of the job name of a background job. The address points to an 8-byte area containing the job name (left-justified and padded with blanks if necessary). The JOBNAME parameter is used during authorization checking to verify the user's authority to submit the job. It is passed to the installation exit routine.

On z/VM, a background job can be used to submit a job to a batch service machine for processing.

,LOG=ASIS
,LOG=ALL
,LOG=NONE
specifies when log records are to be generated.

The default is ASIS.

ASIS
Only those requests to create an ACEE that fail generate RACF log records.

ALL
A request to create an ACEE, regardless of whether it succeeds or fails, generates a RACF log record.

NONE
A request to create an ACEE, regardless of whether it succeeds or fails, does *not* generate a RACF log record.

,LOGSTR=*logstr addr*
specifies the address of a 1-byte length field followed by character data to be written to the system-management-facilities (SMF) data set together with any RACF audit information, if logged.

,NEWPASS=*new password addr*
specifies the password that is to replace the user's currently defined password. The address points to a 1-byte length field, followed by the password, which can be up to eight characters.

The NEWPASS= keyword has no effect unless PASSCHK=YES is either defaulted to or explicitly specified and PASSWRD= is also specified. If the NEWPASS= keyword is specified with PASSCHK=NO, no error message is issued, but the password is not changed. A new password cannot be set when using a password phrase for authentication, nor can a new password phrase be set when using a password for authentication.

Application considerations: When verifying a new password, validate that it is 1-8 characters in length. If SETROPTS MIXEDCASE is not in effect, you must change the password to uppercase. Avoid performing any other checking of character content, letting the security product determine the validity of the password.

,NEWPHRASE=*new password phrase addr*

's currently defined password phrase. The address points to a 1-byte length field, followed by the password phrase, which can be 14-100 characters (or 9-100 characters if ICHPWX11 is present and accepts the new value). If the length byte is 0, the keyword is ignored.

RACF checks the following set of basic rules for the val specifies the password phrase that is to replace the userue specified by NEWPHRASE:

- Must not contain the user ID (as sequential uppercase or sequential lowercase characters)
- Must contain at least 2 alphabetic characters (A-Z, a-z)
- Must contain at least 2 non-alphabetic characters (numerics, punctuation, or special characters)
- Must not contain more than 2 consecutive characters that are identical
- Must not contain forward slashes, nulls (X'00'), or leading or trailing blanks

If NEWPHRASE is specified without PHRASE, it will not be used. A new password phrase cannot be set when using a password for authentication, nor can a new password be set when using a password phrase for authentication.

If NEWPHRASE is specified with PASSCHK=NO, no error message will be issued, but the password phrase will not be changed.

When specifying NEWPHRASE=, you must also specify RELEASE=530 or later.

,PASSCHK=YES
,PASSCHK=NO
,PASSCHK=NOMFA

specifies whether the user's password, password phrase, and/or MFA credentials are to be verified.

YES

RACROUTE REQUEST=VERIFY verifies the user's password, password phrase, and/or MFA credentials. If the user is enabled for MFA, the PASSWRD or PHRASE is exclusively verified as an MFA credential; no other verification methods will be attempted.

There are some circumstances where verification does not occur even though PASSCHK=YES is specified. For examples of surrogate processing, see *z/VM: RACF Security Server Security Administrator's Guide*.

NO

The user's password, password phrase, and/or MFA credentials are not verified and statistics are not updated. And, if the logon is successful, no message is issued.

When PASSCHK=NO is specified, the request will not result in the user being revoked even if the user's statistics have not been updated within *k* days (where *k* is the inactive period defined using SETROPTS INACTIVE(*k*)).

NOMFA

Specifies that the user's password or password phrase be verified as a password or password phrase. The credentials entered are not verified for MFA, even for an MFA enabled user. The application issuing this request should check if the user is PWFALLBACK enabled and if behavior identical to LOGON FALLBACK is desired. Use of the NOMFA parameter requires that RELEASE=1.9 or later be specified.

,PASSWRD=*password addr*
specifies the currently defined password of the user who has entered the system. The address points to a 1-byte length field, followed by the password, which can be up to eight characters.

Application considerations: When verifying a new password, validate that it is 1-8 characters in length. If SETROPTS MIXEDCASE is not in effect, you must change the password to uppercase. Avoid performing any other checking of character content, letting the security product determine the validity of the password.

,PGMNAME=*programmer name addr*
specifies the address of the name of the user who has entered the system. This 20-byte area is passed to the RACINIT installation exit routine; it is not used by the RACROUTE REQUEST=VERIFY routine.

PHRASE=*password phrase addr*
specifies the currently defined password phrase of the user who has entered the system. The address points to a 1-byte length field, followed by the password phrase, which can be 9-100 characters. If the length byte is 0, the keyword is ignored.

The PASSWRD parameter will not be used if the PHRASE parameter is specified with a non-zero length.

Password phrases will not be checked in cases where a password is not checked (PASSCHK=NO specified, or SURROGAT processing).

When specifying PHRASE=, you must also specify RELEASE=530 or later.

,POE=*port of entry addr*
specifies the address of the port of entry into the system. The address points to the name of the input device through which the user or job entered the system. For example, this could be the terminal logged onto. The port of entry is an 8-character field that is left-justified and padded with blanks.

The port of entry becomes a part of the user's security token (UTOKEN). A flag in the UTOKEN uniquely identifies the RACF general-resource class to which the data in the POE field belongs.

The TERMINAL class covers the terminal used to log onto z/VM.

When both the POE and TERMID keywords are specified, the POE keyword takes precedence. Information specified by POE= on an ENVIR=CREATE may be attached to the created ACEE and used in subsequent RACF processing. RACF does not make its own copy of this area when attaching this information to the created ACEE. This area must not be explicitly freed prior to the deletion of the ACEE. For the same reason, the area must reside in a non-task-related storage subpool so that implicit freeing of the area does not occur.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=VERIFY macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Execution-time validation of the compatibility between the list and execute forms of the RACROUTE REQUEST=VERIFY macro can be done by your specifying the CHECK subparameter on the execute form of the macro.

,REMOTE=YES
,REMOTE=NO
specifies whether or not the job came through the network. The default is REMOTE=NO.

,SECLABL=*seclabel addr*
specifies the address of an 8-byte, left-justified character field containing the security label, padded to the right with blanks.

To use this keyword, you must also specify RELEASE=1.9 or a later release number.

An installation may use security labels to establish an association between a specific RACF security level (SECLEVEL) and a set of (zero or more) RACF security categories (CATEGORY). If it is necessary to use security labels to prevent the unauthorized movement of data from one level to another when multiple levels of data are in use on the system at the same time, see *z/VM: RACF Security Server Security Administrator's Guide* for further information.

,SESSION=*type*
specifies the session type to be associated with the request. Session types are literals. When the SESSION keyword is used in combination with the POE keyword, SESSION determines the class with which the POE keyword will be connected.

The default session type is TSO. This is the only valid session type. This session type refers to any interactive session, such as z/VM logon.

,SGROUP=*submitting group addr*
specifies the address of an area that contains a 1-byte length field followed by the group ID of the user who submitted the unit of work. The group ID cannot exceed eight bytes.

,SNODE=*submitting node addr*
specifies the address of an area that contains a 1-byte length field followed by the name of the node from which the unit of work was submitted. The node name cannot exceed eight bytes.

,STAT=ASIS
,STAT=NO
specifies whether the statistics controlled by the options specified on the RACF SETROPTS command should be maintained or ignored for this execution of RACROUTE REQUEST=VERIFY. This parameter also controls whether a message is to be issued when the logon is successful.

The default is ASIS.

Note: Messages are always issued if the RACROUTE REQUEST=VERIFY processing is unsuccessful.

ASIS
The messages and statistics are controlled by the installation's current options on the RACF SETROPTS command.

NO
The statistics are not updated. And, if the logon is successful, no message is issued.

When STAT=NO is specified, the request will not result in the user being revoked even if the user's statistics have not been updated within *k* days (where *k* is the inactive period defined using SETROPTS INACTIVE(*k*)).

,STOKEN=*stoken addr*
specifies the address of the submitter's UTOKEN. The first byte contains the length of the UTOKEN, and the second byte contains the format version number. See ICHRUTKN mapping, “RUTKN” on page 421 for the current version and release.

If you specify an STOKEN, the USERID in the STOKEN becomes the submitter's ID in the ACEE's token unless you specified the submitter's ID (SUSER) keyword. If you did, that keyword becomes the submitter's ID in the ACEE's token. Likewise, if you specified GROUP in STOKEN, that becomes the submitter's group in the ACEE's token, unless you specified the submitter's group (SGROUP) keyword. The SESSION, port-of-entry (POE), and port-of-entry class (POEX) fields are also used from the STOKEN. The execution node becomes the resulting submit node and execution node, unless you specify the submit node (SNODE) or execution-node address (EXENODE) keywords. In all cases, the specified keywords on the request override the fields of the STOKEN, if one is specified.

Also, STOKEN is used for surrogate checking, security-label dominance, or JESJOBS checking unless different submitter-checking information is supplied.

,SUBPOOL=*subpool number*
specifies the storage subpool from which the ACEE and related storage are obtained. The value of the subpool can be literally specified or passed through a register. When using a register, the subpool number is the value of the least significant byte in the register.

On z/VM in the CMS Environment:

If this parameter is not specified, it defaults to zero. If you specify a subpool greater than 127, RACF substitutes subpool zero. You must adhere to the subpools supported by the CMS/OS simulation of GETMAIN. For more information, see the *z/VM: CMS Application Development Guide for Assembler*.

On z/VM in the GCS Environment:

If this parameter is not specified, it defaults to 243. If you specify a subpool, you must adhere to the subpools supported by GCS. For more information, see the *z/VM: Group Control System*.

,SUSERID=*submitting userid addr*
specifies the address of an area that contains a 1-byte length field, followed by the user ID of the user who submitted the unit of work. The user ID cannot exceed eight bytes.

,TERMID=*terminal addr*
specifies the address of the identifier for the terminal through which the user is accessing the system. The address points to an 8-byte area containing the terminal identifier. Information specified by TERMID= on an ENVIR=CREATE may be attached to the created ACEE and used in subsequent RACF processing. RACF does not make its own copy of this area when attaching this information to the created ACEE. This area must not be explicitly freed prior to the deletion of the ACEE. For the same reason, the area must reside in a non-task-related storage subpool so that implicit freeing of the area does not occur. If POE= is specified, the TERMID= area is not referred to in subsequent processing and may be freed at the user's discretion.

,TOKNOUT=*utoken addr*
specifies an address that points to a user-provided area in which the UTOKEN will be built. The mapping of the area is a 1-byte length field, followed by a 1-byte version code, followed by a 78-byte area in which to build the UTOKEN. This token is extracted from the ACEE built by this request.

,TOKNIN=*utoken addr*
specifies an address that points to a caller-provided area that contains an input UTOKEN. The mapping of the area is a 1-byte length field, followed by a 1-byte version code, followed by the UTOKEN itself. The TOKNIN should have been previously obtained by RACROUTE REQUEST=VERIFYX, TOKENXTR or TOKENBLD.

,TRUSTED=YES
,TRUSTED=NO
specifies whether the unit of work is a member of the trusted computer base. Subsequent RACROUTE REQUEST=AUTH requests using an ACEE with this attribute (or a token extracted from the ACEE) have the following effects:

- Authorization checking is bypassed (this includes bypassing the checks for security classification on users and data)
- No statistics are updated
- No audit records are generated, except those requested using the SETROPTS LOGOPTIONS command.

,USERID=*userid addr*
specifies the user identification of the user who has entered the system. The address points to a 1-byte length field, followed by the user ID, which can be up to eight characters.

Application considerations: When verifying a user ID, be sure to validate that it contains only characters that are alphabetic, numeric, # (X'7B'), @ (X'7C'), or $ (X'5B') and is 1-8 characters in length. Additionally, you must change the user ID to uppercase.

,MF=S
specifies the standard form of the RACROUTE REQUEST=VERIFY macro instruction.

Special Considerations for Changing or Deleting an ACEE

IBM recommends that you delete only an ACEE that you created. Issuing a RACROUTE REQUEST=VERIFY with ENVIR=DELETE specified to delete the existing ACEE can lead to problems if you were not the one who created that environment. The issuer of the ENVIR=CREATE that built the ACEE may have saved a pointer to it and may be expecting it to still be available later in processing. Note that this is the case for the initiator's ACEE. Also, if you delete an ACEE, you may lose tables anchored off that ACEE that are needed later in RACF processing. See *z/VM: RACF Security Server Diagnosis Guide* for a overview diagrams of ACEEs and related control blocks. These diagrams can be useful when diagnosing problems.

Note: When you delete an ACEE that has a third-party ACEE attached, the RACINIT pre- or post-exits get control again for the third-party ACEE as well as for the original ACEE being deleted.

If you make a copy of the ACEE and update fields, avoid passing it to RACF. Many RACF services anchor tables off the ACEE and refresh these tables when required. If you update fields in a copy, the original ACEE contains invalid pointers that result in abends when the original is used or deleted. In general, it is recommended that you do not copy an ACEE.

If you need to delete or change an ACEE that you did not create, you can use one of the following methods.

- **Change the values in the current ACEE:**
  1. Issue RACROUTE REQUEST=VERIFY with ENVIR=CHANGE to change the values in the current ACEE.
- **Create, anchor, and delete a third-party ACEE:**
  1. Issue RACROUTE REQUEST=AUTH with USERID= and GROUPID=, causing RACF to create, anchor, and delete a third-party ACEE internally.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note: Some RACROUTE parameter errors result in a RACF abend. On z/VM, these abends are simulated by RACF return and reason codes. For RACROUTE REQUEST=VERIFY, RACF return codes 283 and 9C7 correspond to RACF abends, which are documented in *z/VM: RACF Security Server Command Language Reference*. The reason code also reflects the abend reason code.

Note to reader:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=VERIFY has completed successfully.

RACF Return Code
Meaning

00
Indicates a normal completion.

04
Verify token information.

Reason Code
Meaning

0C
Indicates a TOKNIN was specified, but its length was too large.

10
Indicates an STOKEN was specified, but its length was too large.

04
Requested function could not be completed. No RACF decision.

RACF Return Code
Meaning

00
ENVIR=VERIFY was specified without SAF installation exit processing.

04
The user profile is not defined to RACF.

20
RACF is not active.

58
RJE or NJE operator FACILITY class profile not found.

08
Requested function has failed.

RACF Return Code
Meaning

08
The password, password phrase, or MFA credential is not authorized. This return code is also returned when the MFA server is unavailable for MFA enabled users and PASSCK=NOMFA is not being used.

0C
The password or password phrase has expired.

10
The new password or password phrase is not valid.

Reason Code
Meaning

04
An insufficient number of days has passed since the last password or password phrase change.

14
The user is not defined to the group.

18
RACROUTE REQUEST=VERIFY was failed by the installation exit routine.

1C
The user's access has been revoked.

24
The user's access to the specified group has been revoked.

30
The user is not authorized to the port of entry in the TERMINAL class.

Reason Code
Meaning

00
Indicates the user is not authorized to the port of entry.

04
Indicates the user is not authorized to access the system on this day, or at this time of day.

08
Indicates the port of entry may not be used on this day, or at this time of day.

34
The user is not authorized to use the application.

38
SECLABEL checking failed.

Reason Code
Meaning

04
MLACTIVE requires a SECLABEL; none was specified.

08
Indicates the user is not authorized to the SECLABEL.

0C
The system was in a multilevel secure status, and the dominance check failed.

10
Neither the user's nor the submitter's SECLABELs dominate. They are disjoint.

44
A default token is used as input token.

48
Indicates that an unprivileged user issued a RACROUTE REQUEST=VERIFY in a tranquil state (MLQUIET).

4C
NODES checking failed.

Reason Code
Meaning

00
Submitter's node is not allowed access to execution node.

04
NJE failure: UACC of NONE for USERID type of NODES profile.

08
NJE failure: UACC of NONE for GROUP type of NODES profile.

0C
NJE failure: UACC of NONE for SECLABEL type of NODES profile.

10
NJE failure: No local submit node specified.

14
NJE failure: Reverification of translated values failed.

50
Indicates that a surrogate submit attempt failed.

Reason Code
Meaning

04
Indicates the SURROGAT class was inactive.

08
Indicates the submitter is not permitted by the user's SURROGAT class profile.

0C
Indicates that the submitter is not authorized to the SECLABEL under which the job is to run.

54
Indicates that a JESJOBS check failed.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=VERIFY macro; however, the list form of the macro does not have the same release parameter. Macro processing terminates.

Example 1

Operation: Use the standard form of the macro to do the following:

- Create an ACEE for the user ID and its default group.
- Create an ACEE for the user and group and put its address in ACEEANCH.
- Verify that the user named USERNAME is a valid user.
- Verify that the password called PASSWORD is valid.

```
RACROUTE REQUEST=VERIFY ENVIR=CREATE,USERID=USERNAME,   X
         PASSWRD=PASSWORD,ACEE=ACEEANCH
```

Example 2

Operation: Use the standard form to do the following:

- Verify that the user named USERNAME is a valid user.
- Verify that the group named GROUPNAM is a valid group.
- Verify that USERNAME is defined to the group.
- Create an ACEE for the user and group and put its address in ACEEANCH.
- Specify that the user's password is not required.

```
RACROUTE REQUEST=VERIFY,ENVIR=CREATE,USERID=USERNAME,   X
         GROUP=GROUPNAM,ACEE=ACEEANCH,                  X
         PASSCHK=NO
```

RACROUTE REQUEST=VERIFY (List Form)

The list form of the RACROUTE REQUEST=VERIFY macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=VERIFY macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name — *name*: Symbol. Begin *name* in column 1.

␣ — One or more blanks must precede RACROUTE.

RACROUTE

␣ — One or more blanks must follow RACROUTE.

REQUEST=VERIFY

,ACEE=*acee addr*	*acee addr*: A-type address
,ACTINFO=*account addr*	*account addr*: A-type address
,APPL='*applname*'	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address
,ENCRYPT=YES	**Default:** ENCRYPT=YES
,ENCRYPT=NO	
,ENVIR=CREATE	**Default:** ENVIR=CREATE
,ENVIR=VERIFY	
,ENVIR=CHANGE	
,ENVIR=DELETE	
,EXENODE=*execution node addr*	*execution node addr*: A-type address
,GROUP=*group addr*	*group addr*: A-type address
,INSTLN=*parm list addr*	*parm list addr*: A-type address
,JOBNAME=*jobname addr*	*jobname addr*: A-type address
,LOG=ASIS	**Default:** LOG=ASIS
,LOG=ALL	
,LOG=NONE	
,LOGSTR=*logstr addr*	*logstr addr*: A-type address
,NEWPASS=*new password addr*	*new password addr*: A-type address
,NEWPHRASE=*new password phrase addr*	*new password phrase addr*: A-type address
,PASSCHK=YES	**Default:** PASSCHK=YES
,PASSCHK=NO	

,PASSCHK=NOMFA

,PASSWRD=*password addr*
: *password addr*: A-type address

,PGMNAME=*programmer name addr*
: *programmer name addr*: A-type address

,PHRASE=*password phrase addr*
: *password phrase addr*: A-type address

,POE=*port of entry addr*
: *port of entry addr*: A-type address

,RELEASE=*number*
: *number*: See Standard Form

  Default: RELEASE=1.6

,REMOTE=YES

,REMOTE=NO
: **Default:** REMOTE=NO

,SECLABL=*seclabel addr*
: *seclabel addr*: A-type address

,SESSION=*type*
: **Default:** SESSION=TSO

,SGROUP=*submitting group addr*
: *submitting group addr*: A-type address

,SNODE=*submitting node addr*
: *submitting node addr*: A-type address

,STAT=ASIS
: **Default:** STAT=ASIS

,STAT=NO

,STOKEN=*stoken addr*
: *stoken addr*: A-type address

,SUBPOOL=*subpool number*
: **Default:** See explanation of SUBPOOL keyword in "RACROUTE REQUEST=EXTRACT (Standard Form)" on page 93.

,SUSERID=*submitting userid addr*
: *submitting user ID addr*: A-type address

,TERMID=*terminal addr*
: *terminal addr*: A-type address

,TOKNIN=*utoken addr*	*utoken addr*: A-type address
,TOKNOUT=*utoken addr*	*utoken addr*: A-type address
,USERID=*userid addr*	*userid addr*: A-type address
,TRUSTED=YES	
,TRUSTED=NO	**Default:** TRUSTED=NO
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=VERIFY macro instruction with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=VERIFY macro instruction.

RACROUTE REQUEST=VERIFY (Execute Form)

The execute form of the RACROUTE REQUEST=VERIFY macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=VERIFY macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.
RACROUTE	
␣	One or more blanks must follow RACROUTE.
REQUEST=VERIFY	
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,ACTINFO=*account addr*	*account addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,ENCRYPT=YES	

,ENCRYPT=NO

,ENVIR=CREATE
,ENVIR=VERIFY
,ENVIR=CHANGE
,ENVIR=DELETE

,EXENODE=*execution node addr*	*execution node addr*: Rx-type address or register (2) - (12)
,GROUP=*group addr*	*group addr*: Rx-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,JOBNAME=*jobname addr*	*jobname addr*: Rx-type address or register (2) - (12)

,LOG=ASIS
,LOG=ALL
,LOG=NONE

,LOGSTR=*logstr addr*	*logstr addr*: Rx-type address or register (2) - (12)
,NEWPASS=*new password addr*	*new password addr*: Rx-type address or register (2) - (12)
,NEWPHRASE=*new password phrase addr*	*new password phrase addr*: A-type address or register (2) - (12)

,PASSCHK=YES
,PASSCHK=NO
,PASSCHK=NOMFA

,PASSWRD=*password addr*	*password addr*: Rx-type address or register (2) - (12)
,PGMNAME=*programmer name addr*	(2) - (12)
,PHRASE=*password phrase addr*	*password phrase addr*: A-type address or register (2) - (12)

,POE=*port of entry addr* — *port of entry addr*: Rx-type address or register (2) - (12)

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
— *number*: See Standard Form
Default: RELEASE=1.6

,REMOTE=YES
,REMOTE=NO

,SECLABL=*seclabel addr* — *seclabel addr*: Rx-type address or register (2) - (12)

,SESSION=*type* — *type*: Any valid session type

,SNODE=*submitting node addr* — *submitting node addr*: Rx-type address or register (2) - (12)

,SGROUP=*submitting group addr* — *submitting group addr*: Rx-type address or register (2) - (12)

,STAT=ASIS
,STAT=NO

,STOKEN=*stoken addr* — *stoken addr*: Rx-type address or register (2) - (12)

,SUBPOOL=*subpool number* — *subpool number*: Decimal digit 0-255

,SUSERID=*submitting userid addr* — *submitting userid addr*: Rx-type address or register (2) - (12)

,TERMID=*terminal addr* — *terminal addr*: Rx-type address or register (2) - (12)

,TOKNIN=*utoken addr* — *utoken addr*: Rx-type address or register (2) - (12)

,TOKNOUT=*utoken addr* — *utoken addr*: Rx-type address or register (2) - (12)

,TRUSTED=YES
,TRUSTED=NO

,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=VERIFY macro instruction with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=VERIFY macro, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=VERIFY macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACROUTE REQUEST=VERIFY macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACROUTE REQUEST=VERIFY (Modify Form)

The modify form of the RACROUTE REQUEST=VERIFY macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=VERIFY macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=VERIFY	
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)

,ACTINFO=*account addr* — *account addr*: Rx-type address or register (2) - (12)

,APPL=*applname addr* — *applname addr*: Rx-type address or register (2) - (12)

,ENCRYPT=YES
,ENCRYPT=NO

,ENVIR=CREATE
,ENVIR=VERIFY
,ENVIR=CHANGE
,ENVIR=DELETE

,EXENODE=*execution node addr* — *execution node addr*: Rx-type address or register (2) - (12)

,GROUP=*group addr* — *group addr*: Rx-type address or register (2) - (12)

,INSTLN=*parm list addr* — *parm list addr*: Rx-type address or register (2) - (12)

,JOBNAME=*jobname addr* — *jobname addr*: Rx-type address or register (2) - (12)

,LOG=ASIS
,LOG=ALL
,LOG=NONE

,LOGSTR=*logstr addr* — *logstr addr*: Rx-type address or register (2) - (12)

,NEWPASS=*new password addr* — *new password addr*: Rx-type address or register (2) - (12)

,NEWPHRASE=*new password phrase addr* — *new password phrase addr*: A-type address or register (2) - (12)

,PASSCHK=YES
,PASSCHK=NO
,PASSCHK=NOMFA

,PASSWRD=*password addr* — *password addr*: Rx-type address or register (2) - (12)

,PGMNAME=*programmer name addr* — *programmer name addr*: Rx-type address or register (2) - (12)

,PHRASE=*password phrase addr* — *password phrase addr*: A-type address or register (2) - (12)

,POE=*port of entry addr* — *port of entry addr*: Rx-type address or register (2) - (12)

,RELEASE=*number* — *number*: See Standard Form
Default: RELEASE=1.6

,REMOTE=YES
,REMOTE=NO

,SECLABL=*seclabel addr* — *seclabel addr*: Rx-type address or register (2) - (12)

,SESSION=*type* — *type*: Any valid session type

,SGROUP=*submitting group addr* — *submitting group addr*: Rx-type address or register (2) - (12)

,SNODE=*submitting node addr* — *submitting node addr*: Rx-type address or register (2) - (12)

,STAT=ASIS
,STAT=NO

,STOKEN=*stoken addr* — *stoken addr*: Rx-type address or register (2) - (12)

,SUBPOOL=*subpool number* — *subpool number*: Decimal digit 0-255

,SUSERID=*submitting userid addr* — *submitting userid addr*: Rx-type address or register (2) - (12)

,TERMID=*terminal addr* — *terminal addr*: Rx-type address or register (2) - (12)

,TOKNIN=*utoken addr* — *utoken addr*: Rx-type address or register (2) - (12)

,TOKNOUT=*utoken addr*	*utoken addr*: Rx-type address or register (2) - (12)
,TRUSTED=YES ,TRUSTED=NO	
,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,MF=(M,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=VERIFY macro instruction with the following exception:

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=VERIFY macro, using a remote, control-program parameter list.

RACROUTE REQUEST=VERIFYX: Verify User and Return a UTOKEN

The RACROUTE REQUEST=VERIFYX macro verifies a user and builds a UTOKEN based on the information passed in the parameter list, and handles the propagation of submitter ID.

If the caller specifies an already-existing STOKEN to VERIFYX, and if the caller additionally specifies any UTOKEN keywords on the request, the UTOKEN keywords that are specified override the corresponding parameters in the STOKEN that was passed. Thus, if the caller specifies an STOKEN, the caller should not specify any additional parameters unless the caller wants to supplement STOKEN information.

The following order of priority exists for replacing the fields in the existing TOKEN:

- Keywords specified on the request take precedence over corresponding fields in the TOKNIN and STOKEN parameters.
- All fields within the token specified by the TOKNIN keyword take precedence over those specified by STOKEN.
- The fields for the submitter's ID, submitter's group, submit node, execution node, session, port of entry and its class, as obtained from the token specified by the STOKEN keyword are last.

If you do not want certain fields overridden, do not specify keywords for those fields.

To use this service, you must also specify RELEASE=1.9 or a later release number.

When RACF is installed, the caller of RACROUTE REQUEST=VERIFYX must have at least UPDATE authority to the ICHCONN profile in the FACILITY class. For details on the ICHCONN profile, see "Authorization to Issue RACROUTE Requests" on page 12.

RACROUTE REQUEST=VERIFYX (Standard Form)

The standard form of the RACROUTE REQUEST=VERIFYX macro is written as follows. For a description of additional keywords that you can code and additional parameters that are required on the RACROUTE request, but that are not specific to this request type, see the standard form of the RACROUTE macro.

name	*name*: Symbol. Begin *name* in column 1.

␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=VERIFYX	
,RELEASE=*number*	*number*: 1.9.2 or 1.9
,TOKNOUT=*utoken addr*	*utoken addr*: A-type address or register (2) - (12)
,ACTINFO=*account addr*	*account addr*: A-type address or register (2) - (12)
,APPL=*'applname'* ,APPL=*applname addr*	*applname*: 1- to 8-character name *applname addr*: A-type address or register (2) - (12)
,ENCRYPT=YES ,ENCRYPT=NO	**Default:** ENCRYPT=YES
,EXENODE=*execution node addr*	*execution node addr*: A-type address or register (2) - (12)
,GROUP=*group addr*	*group addr*: A-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12)
,JOBNAME=*jobname addr*	*jobname addr*: A-type address or register (2) - (12)
,LOG=ALL ,LOG=ASIS ,LOG=NONE	**Default:** LOG=ASIS
,LOGSTR=*logstr addr*	*logstr addr*: A-type address or register (2) - (12)
,NEWPASS=*new password addr*	*new password addr*: A-type address or register (2) - (12)
,NEWPHRASE=*new password phrase addr*	*new password phrase addr*: A-type address or register (2) - (12)

,PASSCHK=YES ,PASSCHK=NO ,PASSCHK=NOMFA	**Default:** PASSCHK=YES
,PASSWRD=*password addr*	*password addr*: A-type address or register (2) - (12)
,PGMNAME=*programmer name addr*	*programmer name addr*: A-type address or register (2) - (12)
,PHRASE=*password phrase addr*	*password phrase addr*: A-type address or register (2) - (12)
,POE=*port of entry addr*	*port of entry addr*: A-type address or register (2) - (12)
,REMOTE=YES ,REMOTE=NO	**Default:** REMOTE=NO
,SECLABL=*seclabel addr*	*seclabel addr*: A-type address or register (2) - (12)
,SESSION=*type*	**Default:** SESSION=TSO
,SGROUP=*submitting group addr*	*submitting group addr*: A-type address or register (2) - (12)
,SNODE=*submitting node addr*	*submitting node addr*: A-type address or register (2) - (12)
,STAT=ASIS ,STAT=NO	**Default:** STAT=ASIS
,STOKEN=*stoken addr*	*stoken addr*: A-type address or register (2) - (12)
,SUSERID=*submitting userid addr*	*submitting userid addr*: A-type address or register (2) - (12)
,TERMID=*terminal addr*	*terminal addr*: A-type address or register (2) - (12)
,TOKNIN=*utoken addr*	*utoken addr*: A-type address or register (2) - (12)
,TOKNOUT=*utoken addr*	*utoken addr*: A-type address or register (2) - (12)

,TRUSTED=YES

,TRUSTED=NO **Default:** TRUSTED=NO

,USERID=*userid addr* *userid addr*: A-type address or register (2) - (12)

,MF=S

The parameters are explained as follows:

,ACTINFO=*account addr*
specifies the address of a field containing accounting information. This 144-byte area is passed to the RACINIT installation exit routine; it is not used by the RACROUTE REQUEST=VERIFY routine. The accounting field, if supplied, should have the following format:

- The first byte of the field contains the number (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.

,APPL=*'applname'*
,APPL=*applname addr*
specifies the name of the application issuing the RACROUTE REQUEST=VERIFYX. If an address is specified, the address must point to an 8-byte application name, left justified and padded with blanks if necessary.

,ENCRYPT=YES
,ENCRYPT=NO
specifies whether RACROUTE REQUEST=VERIFYX encodes the old password and the new password passed to it.

The default is YES.

YES
Data specified by the PASSWRD and NEWPASS keywords are not pre-encoded. RACROUTE REQUEST=VERIFYX encodes the data before storing it in the user profile or using it to compare against stored data.

NO
Data specified by the PASSWRD and NEWPASS keywords is already encoded. RACROUTE REQUEST=VERIFYX bypasses the encoding of this data before storing it in, or comparing it against, the user profile.

Notes:

1. ENCRYPT=NO does not apply to PHRASE and NEWPHRASE and will be ignored if specified.
2. If a RACF password is encrypted using KDFAES, then the data that is specified by the PASSWRD= keyword must be encoded using the DES method to be evaluated successfully. If SETROPTS PASSWORD(ALGORITHM(KDFAES)) is active, then the data that is specified by the NEWPASS= keyword must be encoded using the DES method to create a new password that is correctly evaluated.

,EXENODE=*execution node addr*
specifies the address of an area that contains a 1-byte length field followed by the name of the node on which the unit of work is to be executed. The node name cannot exceed eight bytes.

,GROUP=*group addr*
specifies the group of the user who has entered the system. The address points to a 1-byte length field, followed by the group name, which can be up to eight characters long.

,INSTLN=*parm list addr*
specifies the address of an area containing parameter information meaningful to the RACINIT installation exit routine. This area is passed to the installation exit when the exit routine is given control from the RACROUTE REQUEST=VERIFY routine.

The INSTLN parameter can be used by an installation having a user verification or job initiation application, and wanting to pass information from one installation module to the installation exit routine.

The address must point to a 1-byte length field, followed by the parameter list. Note that the parameter list must not contain any address.

,JOBNAME=*jobname addr*
specifies the address of the job name of a background job. The address points to an 8-byte area containing the job name (left-justified and padded with blanks if necessary).

On z/VM, a background job may be used to submit a job to a batch service machine for processing.

Note: The JOBNAME parameter is used by RACF during RACROUTE REQUEST=VERIFYX authorization checking to verify the user's authority to submit the job. It is also passed to the installation RACINIT exit routine.

,LOG=ALL
,LOG=ASIS
,LOG=NONE
specifies when log records are to be generated.

The default is LOG=ASIS.

ALL
Any request to create an ACEE, regardless of whether it succeeds or fails, generates a RACF log record.

ASIS
Only those attempts to create an ACEE that fail generate RACF log records.

NONE
A request to create an ACEE, regardless of whether it succeeds or fails, does *not* generate a RACF log record.

,LOGSTR=*logstr addr*
specifies the address of a 1-byte length field followed by character data that is written to the SMF data set, together with RACF audit information.

,NEWPASS=*new password addr*
specifies the password to replace the user's currently defined password. The address points to a 1-byte length field, followed by the password, which can be up to eight characters long.

The NEWPASS= keyword has no effect unless PASSCHK=YES is either defaulted to or explicitly specified and PASSWRD= is also specified. If the NEWPASS= keyword is specified with PASSCHK=NO, no error message is issued, but the password is not changed. A new password cannot be set when using a password phrase for authentication, nor can a new password phrase be set when using a password for authentication.

Application considerations: When verifying a new password, validate that it is 1-8 characters in length. If SETROPTS MIXEDCASE is not in effect, you must change the password to uppercase. Avoid performing any other checking of character content, letting the security product determine the validity of the password.

,NEWPHRASE=*new password phrase addr*
specifies the password phrase that is to replace the user's currently defined password phrase. The address points to a 1-byte length field, followed by the password phrase, which can be 14-100 characters (or 9-100 characters if ICHPWX11 is present and accepts the new value). If the length byte is 0, the keyword is ignored.

RACF checks the following set of basic rules for the value specified by NEWPHRASE:

- Must not contain the user ID (as sequential uppercase or sequential lowercase characters)
- Must contain at least 2 alphabetic characters (A-Z, a-z)
- Must contain at least 2 non-alphabetic characters (numerics, punctuation, or special characters)
- Must not contain more than 2 consecutive characters that are identical
- Must not contain forward slashes, nulls (X'00'), or leading or trailing blanks

If NEWPHRASE is specified without PHRASE, it will not be used. A new password phrase cannot be set when using a password for authentication, nor can a new password be set when using a password phrase for authentication.

If NEWPHRASE is specified with PASSCHK=NO, no error message will be issued, but the password phrase will not be changed.

When specifying NEWPHRASE=, you must also specify RELEASE=530 or later.

,PASSCHK=YES
,PASSCHK=NO
,PASSCHK=NOMFA
specifies whether or not the user's password, password phrase, and/or MFA credentials are to be verified.

YES
RACROUTE REQUEST=VERIFY verifies the user's password, password phrase, and/or MFA credentials. If the user is enabled for MFA, the PASSWRD or PHRASE is exclusively verified as an MFA credential; no other verification methods will be attempted.

NO
The user's password, password phrase, and/or MFA credentials are not verified and statistics are not updated. And, if the logon is successful, no message is issued.

When PASSCHK=NO is specified, the request will not result in the user being revoked even if the user's statistics have not been updated within *k* days (where *k* is the inactive period defined using SETROPTS INACTIVE(*k*)).

NOMFA
Specifies that the user's password or password phrase be verified as a password or password phrase. The credentials entered are not verified for MFA, even for an MFA enabled user.
The application issuing this request should check if the user is PWFALLBACK enabled and if behavior identical to LOGON FALLBACK is desired. Use of the NOMFA parameter requires that RELEASE=1.9 or later be specified.

,PASSWRD=*password addr*
specifies the currently defined password of the user who has entered the system. The address points to a 1-byte length field, followed by the password, which can be up to eight characters long.

Application considerations: When verifying a new password, validate that it is 1-8 characters in length. If SETROPTS MIXEDCASE is not in effect, you must change the password to uppercase. Avoid performing any other checking of character content, letting the security product determine the validity of the password.

,PGMNAME=*programmer name addr*
specifies the address of the name of the user who has entered the system. This 20-byte area is passed to the RACINIT installation exit routine; it is not used by RACF.

PHRASE=*password phrase addr*
specified the currently defined password phrase of the user who has entered the system. The address points to a 1-byte length field, followed by the password phrase, which can be 9-100 characters. If the length byte is 0, the keyword is ignored.

The PASSWRD parameter will not be used if the PHRASE parameter is specified with a non-zero length.

Password phrases will not be checked in cases where a password is not checked (PASSCHK=NO specified, or SURROGAT processing).

When specifying PHRASE=, you must also specify RELEASE=530 or later.

,POE=*port of entry addr*
specifies the address of the port of entry into the system. The address points to the name of the job entered the system. For example, this could be the name of the terminal logged onto. The port of entry is an 8-character field that is left-justified and padded with blanks.

The port of entry becomes a part of the user's security token (UTOKEN). A flag in the UTOKEN uniquely identifies the RACF general-resource class to which the data in the POE field belongs.

The TERMINAL class covers the terminal used to log onto z/VM.

When both the POE and TERMID keywords are specified, the POE keyword takes precedence.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=VERIFYX macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. When you specify the RELEASE keyword, checking is done at assembly time.

The release specified must be 1.9 or later.

,REMOTE=YES
,REMOTE=NO
specifies whether or not the job came through the network. The default is REMOTE=NO.

,SECLABL=*seclabel addr*
specifies the address of an 8-byte, left-justified character field containing the security label, padded to the right with blanks.

An installation may use security labels to establish an association between a specific RACF security level (SECLEVEL) and a set of (zero or more) RACF security categories (CATEGORY). If it is necessary to use security labels to prevent the unauthorized movement of data from one level to another when multiple levels of data are in use on the system at the same time, see *z/VM: RACF Security Server Security Administrator's Guide* for further information.

,SESSION=*type*
specifies the session type to be associated with the request. Session types are literals. When the SESSION keyword is used in combination with the POE keyword, SESSION determines the class with which the POE keyword will be connected.

The default session type is TSO. This is the only valid session type. This session type refers to any interactive session, such as VM logon.

,SGROUP=*submitting group addr*
specifies the address of an area that contains a 1-byte length field followed by the group ID of the user who submitted the unit of work. The group ID cannot exceed eight bytes.

,SNODE=*submitting node addr*
specifies the address of an area that contains a 1-byte length field, followed by the name of the node from which the unit of work was submitted. The node name cannot exceed eight bytes.

,STAT=ASIS
,STAT=NO
specifies that no statistics will be updated for this execution of RACROUTE REQUEST=VERIFYX, and that if logon is successful, no message will be issued.

When STAT=NO is specified, the request will not result in the user being revoked even if the user's statistics have not been updated within *k* days (where *k* is the inactive period defined using SETROPTS INACTIVE(*k*)).

Note:

1. The default (STAT=ASIS) is processed the same as STAT=NO.
2. Messages are always issued if the RACROUTE REQUEST=VERIFYX processing is unsuccessful.

,STOKEN=*stoken addr*
specifies the address of the submitter's security token (UTOKEN). The first byte contains the length of the UTOKEN, and the second byte contains the format version number. See ICHRUTKN mapping, “RUTKN” on page 421 for the current version and release.

If you specify STOKEN, the user ID in STOKEN becomes the submitter's ID in TOKNOUT, unless you specified the submitter's ID (SUSER) keyword. If you did, that keyword becomes the submitter's ID in TOKNOUT. Likewise, if you specified GROUP in the STOKEN, that becomes the submitter's group in TOKNOUT, unless you specified the submitter's group (SGROUP) keyword. The SESSION, port-of-entry (POE), and port-of-entry class (POEX) fields are also used from the STOKEN. The execution node becomes the resulting submit node and execution node unless you specify the submit node (SNODE) or execution node (EXENODE) keywords. In all cases, the specified keywords on the request override the fields of the STOKEN, if one is specified.

Also, STOKEN is used unless different submitter-checking information, such as surrogate checking, security-label dominance, or JESJOBS checking is specified.

,SUSERID=*submitting userid addr*
specifies the address of an area that contains a 1-byte length field followed by the user ID of the user who submitted the unit of work. The user ID cannot exceed eight bytes.

,TERMID=*terminal addr*
specifies the address of the identifier for the terminal through which the user is accessing the system. The address points to an 8-byte area containing the terminal identifier. The area must reside in a storage subpool not related to any task.

,TOKNIN=*utoken addr*
specifies an address that points to a caller-provided area that contains an input UTOKEN. The mapping of the area is a 1-byte length field, followed by a 1-byte version code, followed by the UTOKEN itself, which can be 78 bytes long. The TOKNIN should have been obtained earlier by RACROUTE REQUEST=VERIFYX, RACROUTE REQUEST=TOKENXTR, or RACROUTE REQUEST=TOKENBLD.

,TOKNOUT=*output token addr*
specifies the address of the caller-provided area in which the UTOKEN will be built. The first byte of storage at the address specified is the token length field. The second byte must contain the format version of the token. It is followed by a 78-byte area in which to build the UTOKEN. The mapping of the area is a 1-byte length field, followed by a 1-byte version code, followed by the rest of the token information.

For a description of the fields TOKNOUT uses from STOKEN, see the STOKEN description.

,TRUSTED=YES
,TRUSTED=NO
specifies whether the unit of work is a member of the trusted computer base. Subsequent RACROUTE REQUEST=AUTH requests using a token with this attribute have the following effects:

- Authorization checking is bypassed (this includes bypassing the checks for security classification on users and data).
- No statistics are updated.
- No audit records are generated, except those requested using the SETROPTS LOGOPTIONS command.

,USERID=*userid addr*
specifies the identification of the user who has entered the system. The address points to a 1-byte length field, followed by the user ID, which can be up to eight characters long.

Application considerations: When verifying a user ID, be sure to validate that it contains only characters that are alphabetic, numeric, # (X'7B'), @ (X'7C'), or $ (X'5B') and is 1-8 characters in length. Additionally, you must change the user ID to uppercase.

,MF=S
specifies the standard form of the RACROUTE REQUEST=VERIFYX macro instruction.

Return Codes and Reason Codes

When you execute the macro, space for the RACF return code and reason code is reserved in the first two words of the RACROUTE parameter list. You can access them using the ICHSAFP mapping macro, by loading the ICHSAFP pointer with the label that you specified on the list form of the macro. When control is returned, register 15 contains the SAF return code.

Note: Some RACROUTE parameter errors result in a RACF abend. On z/VM, these abends are simulated by RACF return and reason codes. For RACROUTE REQUEST=VERIFYX, RACF return codes 283 and 9C7 correspond to RACF abends, which are documented in *z/VM: RACF Security Server Messages and Codes*. The reason code also reflects the abend reason code.

Note to reader:
All return and reason codes are shown in hexadecimal.

SAF Return Code
Meaning

00
RACROUTE REQUEST=VERIFYX has completed successfully.

RACF Return Code
Meaning

00
Indicates a normal completion.

3C
Request completed successfully, but a VERIFYX condition occurred in SAF.

Reason Code
Meaning

20
TOKNOUT area specified was too large; on return, the length field contains the length used.

24
STOKEN area specified was too large.

30
TOKNIN area specified was too large.

04
The requested function could not be performed.

RACF Return Code
Meaning

00
No security decision could be made.

Reason Code
Meaning

00
The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table; no successful exit processing.

20
RACF is not active.

3C
RACF is not installed.

58
RJE or NJE operator FACILITY class profile not found.

08
The requested function failed.

RACF Return Code
Meaning

00
Default ACEE or token-building error.

Reason Code
Meaning

00
SAF failed to set up a recovery environment.

04
The user profile is not defined to RACF.

08
The password, password phrase, or MFA credential is not authorized. This return code is also returned when the MFA server is unavailable for MFA enabled users and PASSCK=NOMFA is not being used.

0C
The password or password phrase has expired.

10
The new password or password phrase is not valid.

Reason Code
Meaning

04
An insufficient number of days has passed since the last password or password phrase change.

14
The user is not defined to the group.

18
RACROUTE REQUEST=VERIFYX was failed by the installation exit routine.

1C
The user's access has been revoked.

24
The user's access to the specified group has been revoked.

30
The user is not authorized to the port of entry.

34
The user is not authorized to use the application.

38
SECLABEL checking failed.

Reason Code
Meaning

04
MLACTIVE requires a security label; none was specified.

08
Indicates the user is not authorized to the security label.

0C
The system was in multilevel secure status, and the dominance check failed.

10
Neither the user's nor the submitter's security labels dominate. They are disjoint.

3C
A VERIFYX error occurred in SAF.

Reason Code
Meaning

04
Old password required. Message IRR009I issued.

08
User ID required. Message IRR008I issued.

0C
Propagation checking could not complete. Failed to set up a recovery environment.

44
A default token is used as input token.

48
Indicates that an unprivileged user issued a RACROUTE REQUEST=VERIFYX in a tranquil state (MLQUIET).

4C
NODES checking failed.

Reason Code
Meaning

00
Submitter's node is not allowed access to execution node.

04
NJE failure: UACC of NONE for USERID type of NODES profile.

08
NJE failure: UACC of NONE for GROUP type of NODES profile.

0C
NJE failure: UACC of NONE for SECLABEL type of NODES profile.

10
NJE failure: No local submit node specified.

14
NJE failure: Reverification of translated values failed.

50
Indicates that a surrogate submit attempt failed.

Reason Code
Meaning

04
Indicates the SURROGAT class was inactive.

08
Indicates the submitter is not permitted by the user's SURROGAT class profile.

0C
Indicates that the submitter is not authorized to the security label under which the job is to run.

54
Indicates that a JESJOBS check failed.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACROUTE REQUEST=VERIFYX macro; however, the list form of the macro does not have the same release parameter. Macro processing terminates.

Example 1

Operation: The following example shows a RACROUTE REQUEST=VERIFYX coded for handling verification checking for a batch job that has been submitted with a USERID, GROUPID, SECLABEL, and PASSWORD. The UTOKEN area will be filled with the verified job information.

```
   RACROUTE REQUEST=VERIFYX,SESSION=INTBATCH,          X
            PASSWRD=PASSWORD,TOKNOUT=TOKOUT,           X
            EXENODE=EXNOD,USERID=USER,                 X
            GROUP=GROUPID,SECLABL=SLBL,                X
            STOKEN=STOK,TRUSTED=NO,                    X
            RELEASE=1.9
  .
  .
PASSWORD  DS  0CL9
PASSWL    DS  FL1'5'
PASSWT    DS  CL8'PWD01'

TOKOUT    DS  0CL80
TKOLEN    DS  XL1'50'   /* LENGTH - 80 DEC */
TKOVRS    DS  XL1'01'   /* VERSION 1       */
TKODATA   DS  CL78

EXNOD     DS  0CL9
EXNODL    DS  FL1'2'
EXNODT    DS  CL8'N1'

USER      DS  0CL9
USERL     DS  FL1'6'
USERT     DS  CL8'USER01'

GROUPID   DS  0CL9
GROUPIDL  DS  FL1'4'
GROUPIDT  DS  CL8'SYS1'

SLBL      DS  CL8'SYSLOW'

STOK      DS  CL80          /* OBTAINED BY PREVIOUS RACROUTE CALL */
```

Note: Additional keywords required by RACF to complete the request, such as WORKA, are specified on RACROUTE itself.

RACROUTE REQUEST=VERIFYX (List Form)

The list form of the RACROUTE REQUEST=VERIFYX macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=VERIFYX macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=VERIFYX	
,RELEASE=*number*	*number*: See Standard Form

,ACTINFO=*account addr*	*account addr*: A-type address
,APPL='*applname*'	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address
,ENCRYPT=YES ,ENCRYPT=NO	**Default:** ENCRYPT=YES
,EXENODE=*execution node addr*	*execution node addr*: A-type address
,GROUP=*group addr*	*group addr*: A-type address
,INSTLN=*parm list addr*	*parm list addr*: A-type address
,JOBNAME=*jobname addr*	*jobname addr*: A-type address
,LOG=ASIS ,LOG=ALL	**Default:** LOG=ASIS
,LOGSTR=*logstr addr*	*logstr addr*: A-type address
,NEWPASS=*new password addr*	*new password addr*: A-type address
,NEWPHRASE=*new password phrase addr*	*new password phrase addr*: A-type address or register (2) - (12)
,PASSCHK=YES ,PASSCHK=NO ,PASSCHK=NOMFA	**Default:** PASSCHK=YES
,PASSWRD=*password addr*	*password addr*: A-type address
,PGMNAME=*programmer name addr*	*programmer name addr*: A-type address
,PHRASE=*password phrase addr*	*password phrase addr*: A-type address or register (2) - (12)
,POE=*port of entry addr*	*port of entry addr*: A-type address

,REMOTE=YES	
,REMOTE=NO	**Default:** REMOTE=NO
,SECLABL=*seclabel addr*	*seclabel addr*: A-type address
,SESSION=*type*	**Default:** SESSION=TSO
,SGROUP=*submitting group addr*	*submitting group addr*: A-type address
,SNODE=*submitting node addr*	*submitting node addr*: A-type address
,STAT=ASIS	**Default:** STAT=ASIS
,STAT=NO	
,STOKEN=*stoken addr*	*stoken addr*: A-type address
,SUSERID=*submitting userid addr*	*submitting userid addr*: A-type address
,TERMID=*terminal addr*	*terminal addr*: A-type address
,TOKNIN=*utoken addr*	*utoken addr*: A-type address
,TRUSTED=YES	
,TRUSTED=NO	**Default:** TRUSTED=NO
,USERID=*userid addr*	*userid addr*: A-type address
,MF=L	

The parameters are explained under the standard form of the RACROUTE REQUEST=VERIFYX macro instruction, with the following exception:

,MF=L
specifies the list form of the RACROUTE REQUEST=VERIFYX macro instruction.

RACROUTE REQUEST=VERIFYX (Execute Form)

The execute form of the RACROUTE REQUEST=VERIFYX macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=VERIFYX macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=VERIFYX	
,RELEASE=*number* ,RELEASE=(*number*,CHECK)	*number*: See Standard Form
,ACTINFO=*account addr*	*account addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,ENCRYPT=YES ,ENCRYPT=NO	
,EXENODE=*execution node addr*	*execution node addr*: Rx-type address or register (2) - (12)
,GROUP=*group addr*	*group addr*: Rx-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,JOBNAME=*jobname addr*	*jobname addr*: Rx-type address or register (2) - (12)
,LOG=ASIS ,LOG=ALL	
,LOGSTR=*logstr addr*	*logstr addr*: Rx-type address or register (2) - (12)

,NEWPASS=*new password addr* — *new password addr*: Rx-type address or register (2) - (12)

,NEWPHRASE=*new password phrase addr* — *new password phrase addr*: A-type address or register (2) - (12)

,PASSCHK=YES

,PASSCHK=NO

,PASSCHK=NOMFA

,PASSWRD=*password addr* — *password addr*: Rx-type address or register (2) - (12)

,PGMNAME=*programmer name addr* — *programmer name addr*: Rx-type address or register (2) - (12)

,PHRASE=*password phrase addr* — *password phrase addr*: A-type address or register (2) - (12)

,POE=*port of entry addr* — *port of entry addr*: Rx-type address or register (2) - (12)

,REMOTE=YES

,REMOTE=NO

,SECLABL=*seclabel addr* — *seclabel addr*: Rx-type address or register (2) - (12)

,SESSION=*type* — *type*: Any valid session type

,SGROUP=*submitting group addr* — *submitting group addr*: Rx-type address or register (2) - (12)

,SNODE=*submitting node addr* — *submitting node addr*: Rx-type address or register (2) - (12)

,STAT=ASIS

,STAT=NO

,STOKEN=*stoken addr* — *stoken addr*: Rx-type address or register (2) - (12)

,SUSERID=*submitting userid addr* — *submitting userid addr*: Rx-type address or register (2) - (12)

,TERMID=*terminal addr*	*terminal addr*: Rx-type address or register (2) - (12)
,TOKNIN=*utoken addr*	*utoken addr*: Rx-type address or register (2) - (12)
,TOKNOUT=*utoken addr*	*utoken addr*: Rx-type address or register (2) - (12)
,TRUSTED=YES ,TRUSTED=NO	
,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=VERIFYX macro instruction, with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE REQUEST=VERIFYX macro, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Note: RELEASE=1.10 is not supported on the RACROUTE REQUEST=VERIFYX macro. Invocations of this macro must specify RELEASE=1.9.2 or lower.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

When you specify the RELEASE keyword, checking is done at assembly time. Execution-time validation of the compatibility between the list and execute forms of the RACROUTE REQUEST=VERIFYX macro can be done by your specifying the CHECK subparameter on the execute form of the macro.

When CHECK processing is requested, if the size of the list-form expansion is not large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro, the execute form of the macro is not done.

The release specified must be 1.9 or higher.

RACROUTE REQUEST=VERIFYX (Modify Form)

The modify form of the RACROUTE REQUEST=VERIFYX macro is written as follows. Refer to the Standard Form of the RACROUTE REQUEST=VERIFYX macro to determine additional parameters that are required by the time the RACROUTE service is invoked using the Execute form of the macro.

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACROUTE.

RACROUTE

␣ One or more blanks must follow RACROUTE.

REQUEST=VERIFYX

,RELEASE=*number*
,RELEASE=(*number*,CHECK)
number: See Standard Form

,ACTINFO=*account addr*
account addr: Rx-type address or register (2) - (12)

,APPL=*applname addr*
applname addr: Rx-type address or register (2) - (12)

,ENCRYPT=YES
,ENCRYPT=NO

,EXENODE=*execution node addr*
execution node addr: Rx-type address or register (2) - (12)

,GROUP=*group addr*
group addr: Rx-type address or register (2) - (12)

,INSTLN=*parm list addr*
parm list addr: Rx-type address or register (2) - (12)

,JOBNAME=*jobname addr*
jobname addr: Rx-type address or register (2) - (12)

,LOG=ASIS
,LOG=ALL

,LOGSTR=*logstr addr*
logstr addr: Rx-type address or register (2) - (12)

,NEWPASS=*new password addr*
new password addr: Rx-type address or register (2) - (12)

,NEWPHRASE=*new password phrase addr*
new password phrase addr: A-type address or register (2) - (12)

,PASSCHK=YES
,PASSCHK=NO
,PASSCHK=NOMFA

,PASSWRD=*password addr* — *password addr*: Rx-type address or register (2) - (12)

,PGMNAME=*programmer name addr* — *programmer name addr*: Rx-type address or register (2) - (12)

,PHRASE=*password phrase addr* — *password phrase addr*: A-type address or register (2) - (12)

,POE=*port of entry addr* — *port of entry addr*: Rx-type address or register (2) - (12)

,REMOTE=YES
,REMOTE=NO

,SECLABL=*seclabel addr* — *seclabel addr*: Rx-type address or register (2) - (12)

,SESSION=*type* — *type*: Any valid session type

,SGROUP=*submitting group addr* — *submitting group addr*: Rx-type address or register (2) - (12)

,SNODE=*submitting node addr* — *submitting node addr*: Rx-type address or register (2) - (12)

,STAT=ASIS
,STAT=NO

,STOKEN=*stoken addr* — *stoken addr*: Rx-type address or register (2) - (12)

,SUSERID=*submitting userid addr* — *submitting userid addr*: Rx-type address or register (2) - (12)

,TERMID=*terminal addr* — *terminal addr*: Rx-type address or register (2) - (12)

,TOKNIN=*utoken addr* — *utoken addr*: Rx-type address or register (2) - (12)

,TOKNIN=*utoken addr* — *utoken addr*: Rx-type address or register (2) - (12)

,TRUSTED=YES

,TRUSTED=NO

,USERID=*userid addr* — *userid addr*: Rx-type address or register (2) - (12)

,MF=(M,*ctrl addr*) — *ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACROUTE REQUEST=VERIFYX macro instruction, with the following exception:

,MF=(M,*ctrl addr*)
specifies the modify form of the RACROUTE REQUEST=VERIFYX macro, using a remote, control-program parameter list.

RACSYNC Macro (z/VM Only)

The RACSYNC macro accesses the returned RACROUTE parameter list and loads registers as if the call were synchronous.

Specify the RACSYNC macro after the asynchronous RACROUTE invocation to make it appear as though the request were synchronous.

When control is returned, registers 0, 1, and 15 are set to the RACROUTE reason code, returned data pointer, and RACROUTE return code, respectively. The values set for these items are dependent on the specific RACROUTE invocation specified. For example, on RACROUTE REQUEST=EXTRACT, register 1 may be set to the data area containing information extracted from the RACF database. (See "RACROUTE REQUEST=EXTRACT: Replace or Retrieve Fields" on page 92 for more details.)

The standard format is the only form available for the RACSYNC macro instruction, because it has no need to be reentrant.

The standard form of the RACSYNC macro is written as follows:

name — *name*: Symbol. Begin *name* in column 1.

␣
RACSYNC — One or more blanks must precede RACSYNC.

␣ — One or more blanks must follow RACSYNC.

REQADDR=*racroute request address* — *racroute request address*: A-type address or register (2) - (12)

,REQFORM=S
,REQFORM=L — **Default:** REQFORM=S

,WAIT — **Default:** No Wait

The parameters are explained as follows:

,REQADDR=*racroute request address*
specifies the address of the RACROUTE invocation.

,REQFORM=S
,REQFORM=L
specifies the form of the RACROUTE macro that was invoked.

S
Indicates that the standard form of the macro was specified.

L
Indicates that the list form of the macro was specified.

The execute and modify forms are not indicated, because they both operate on the list form of the macro.

This parameter is optional. REQFORM=S is the default.

,WAIT
specifies that a WAIT is to be issued using the ECB specified on the original RACROUTE macro invocation.

If the WAIT keyword is omitted, RACSYNC obtains addressability to the RACROUTE parameter list and loads registers 0, 1, and 15 from the RACROUTE parameter list. After the RACSYNC invocation, these registers reflect the SAF reason code, the returned data pointer, and the SAF return code, respectively.

If the WAIT keyword is specified, RACSYNC issues a wait using the ECB address that was specified in the original RACROUTE macro instruction. When the request is complete and the ECB is posted, the RACSYNC macro instruction loads the user's registers as described above.

This parameter is optional. The default for this keyword, if it is not specified, is not to wait.

Appendix A. Independent RACF System Macros

CAUTION: The interfaces in this appendix are not recommended for use because they have not been enhanced since Release 1.8.2 and will not be enhanced in future releases. Furthermore, the RACROUTE macros described in Chapter 2, “RACF System Macros,” on page 7, provide more function.

This appendix contains independent RACF system macros that can be used by other callers to invoke RACF or another external security product.

Note: If these macros are used, they can only be invoked from programs that execute within the RACF virtual machine.

As of RACF 1.9, new keywords are not supported on the independent invocation of these macros. RACF supports the new keywords only if you invoke the RACF system macros using the RACROUTE interface documented in Chapter 2, “RACF System Macros,” on page 7.

Table 4 on page 207 identifies the RACROUTE macro request types that are replacements for the independent system macros described in this appendix. If you receive a return code or reason code from an independent system macro and cannot find a description of the code in this appendix, refer to the counterpart request type in RACROUTE.

Table 4. Cross-reference for RACROUTE REQUEST=type and the Independent RACF System Macros

RACROUTE REQUEST Type	replaces	Independent RACF System Macro
REQUEST=AUTH		RACHECK
REQUEST=DEFINE		RACDEF
REQUEST=EXTRACT		RACXTRT
REQUEST=FASTAUTH		FRACHECK
REQUEST=LIST		RACLIST
REQUEST=STAT		RACSTAT
REQUEST=VERIFY		RACINIT

Following is a brief description of the independent RACF system macros.

- **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 macro service.
- **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.
- **RACINIT:** Used to provide RACF user identification and verification.
- **RACLIST:** Used to retrieve general-resource profiles and build an in-storage list for faster authorization checking. The list is attached to the ACEE.
- **RACROUTE on z/VM:** A limited-function RACROUTE used to invoke RACF authorization checking on z/VM.
- **RACSTAT:** Used to determine whether RACF is active, and, optionally, to determine whether RACF protection is in effect for a given resource class. The RACSTAT macro can also be used to determine whether a resource-class name is defined to RACF.
- **RACXTRT:** Used to retrieve or update specified resource-profile fields, or to encode data.

z/VM users receive the RACF system macros as parts of the z/VM product and the RACF product.

© Copyright IBM Corp. 1990, 2022

FRACHECK Macro

The FRACHECK macro is used to check a user's authorization for access to a resource. FRACHECK verifies access to those resources whose RACF profiles have been brought into main storage by the RACLIST facility. FRACHECK is a branch-entered service that does not save registers upon entry. Registers 0-5, 14, and 15 are used by the FRACHECK macro instruction and are not restored. Registers 6-13 are not altered by FRACHECK.

Note:

1. Profile names containing "RACFVARS", double asterisks, or internal asterisks should not be defined in classes that use this macro for authorization. Results are unpredictable.
2. SECLABEL class processing is not done for FRACHECK. It is done for RACROUTE REQUEST=FASTAUTH.

On z/VM, you can use the FRACHECK macro only in the RACF service machine (for example, from an installation exit). You may not use the FRACHECK macro from a user's machine.

FRACHECK (Standard Form)

The standard form of the FRACHECK macro instruction is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ FRACHECK	One or more blanks must precede FRACHECK.
␣	One or more blanks must follow FRACHECK.
ENTITY=*entity addr*	*entity addr*: A-type address or register (2) - (12)
,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER	**Default:** ATTR=READ
,ATTR=*reg*	*reg*: Registers (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12)
,WKAREA=*area addr*	*area addr*: A-type address or register (2) - (12)
,APPL=*'applname'*	*applname*: 1- to 8-character name

,APPL=*applname addr*	*applname addr*: A-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12)
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6

The parameters are explained as follows:

ENTITY=*entity addr*
specifies that RACF authorization checking is to be performed for the resource whose name is pointed to by the specified address. The resource name is a 6-byte volume serial number for CLASS='DASDVOL' or CLASS='TAPEVOL'. The name must be left-justified and padded with blanks. The length of all other resource names is determined from the class-descriptor tables.

,CLASS='*class name*'
,CLASS=*class name addr*
specifies that RACF authorization checking is to be performed for a resource of the specified class. If an address is specified, the address must point to an 8-byte field containing the class name.

,ATTR=READ
,ATTR=UPDATE
,ATTR=CONTROL
,ATTR=ALTER
,ATTR=*reg*
specifies the access authority required by the user or group accessing the resource:

READ
RACF user or group can open the resource only to read.

UPDATE
RACF user or group can open the resource to read or write.

CONTROL
For VSAM data sets, RACF user or group has authority equivalent to the VSAM control password. For non-VSAM data sets and other resources, RACF user or group has UPDATE authority.

ALTER
RACF user or group has total control over the resource.

If a register is specified, the register must contain one of the following codes in the low-order byte of the register:

X'02' READ
X'04' UPDATE
X'08' CONTROL
X'80' ALTER.

,ACEE=*acee addr*
specifies the address of the ACEE to be used to check authorization and to locate the in-storage profiles (RACLIST output) for the specified classes. If an ACEE is specified, it is used for authorization checking. If the specified ACEE has an in-storage profile list for the specified class, it is used to locate the resource. If an ACEE is not specified or if there is no in-storage profile list for the specified class in the ACEE, RACF uses the TASK ACEE (TCBSENV) pointer in the extended TCB. Otherwise, or if the TASK ACEE pointer is zero, RACF uses the main ACEE for the address space to obtain the list of the in-storage profiles. The main ACEE is pointed to by the ASXBSENV field of the address-space extension block.

,WKAREA=*area addr*
specifies the address of a 16-word work area to be used by FRACHECK. It contains the following information:

Word 12 contains the reason code that RACF passes back to the FRACHECK caller in register 0.
Word 13 contains the return code that FRACHECK passes back to the caller in register 15.
Word 14 contains the address of the in-storage profile used to determine authorization, or zero if no profile is found.
Word 15 contains a value provided by a preprocessing installation exit, or zero if there is no preprocessing exit. This is passed back to the caller in register 1.

,APPL=*'applname'*
,APPL=*applname addr*
specifies the name of the application requesting the authorization checking. This information is not used for the authorization-checking process, but is made available to the installation exit or exits. If an address is specified, it should point to an 8-byte area containing the application name, left-justified and padded with blanks if necessary.

,INSTLN=*parm list addr*
specifies the address of an area that contains information for the FRACHECK installation exit. This address is passed to the exit routine when the exit is given control. The INSTLN parameter is used by application or installation programs to pass information to the FRACHECK installation exit.

,RELEASE=1.6|1.7|1.8|1.8.1
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 5 on page 210.

When you specify the RELEASE keyword, checking is done at assembly time. Execution-time validation of the compatibility between the list and execute forms of the FRACHECK macro can be done by your specifying the CHECK subparameter on the execute form of the macro.

Parameters for RELEASE=1.6 through 1.8.1

The RELEASE values for which a specific parameter is valid are marked with an 'X'.

Table 5. FRACHECK Parameters for RELEASE=1.6 through 1.8.1

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
ACEE=	X	X	X
APPL=	X	X	X
ATTR=	X	X	X
CLASS=	X	X	X
ENTITY=	X	X	X
INSTLN=	X	X	X
RELEASE=	X	X	X
WKAREA=	X	X	X

Return Codes and Reason Codes

For FRACHECK, if the return codes and reason codes you are receiving are not discussed in the description of this macro, refer to "Return Codes and Reason Codes" on page 124.

When control is returned, register 15 contains one of the following return codes, and register 0 may contain a reason code:

Hexadecimal
Meaning

00
The user or group is authorized to use the resource.

Reason Code
Meaning

0
The FRACHECK return code indicates whether the caller is authorized to access the resource, but the access attempt is not within the scope of the audit or global audit specification.

4
The FRACHECK return code indicates whether the caller is authorized to access the resource, but the access attempt is within the scope of the audit or global audit specification. The FRACHECK caller should log the attempt by issuing a RACHECK for the resource that the caller is attempting to access.

04
The resource or class name is not defined to RACF.

08
The user or group is not authorized to use the resource.

Reason Code
Meaning

0
The FRACHECK return code indicates whether the caller is authorized to access the resource, but the access attempt is not within the scope of the audit or global audit specification.

4
The FRACHECK return code indicates whether the caller is authorized to access the resource, but the access attempt is within the scope of the audit or global audit specification. The FRACHECK caller should log the attempt by issuing a RACHECK for the resource that the caller is attempting to access.

0C
RACF is not active.

10
FRACHECK installation-exit error occurred.

14
RACF is not installed or an insufficient level of RACF is installed.

18
Indicates the profile has a conditional access list, the port-of-entry field in the security token is blank-filled, and the port-of-entry class is active.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the FRACHECK macro; however, the list form of the macro does not have the proper RELEASE parameter. Macro processing terminates.

FRACHECK (List Form)

The list form of the FRACHECK macro instruction is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ FRACHECK	One or more blanks must precede FRACHECK.
␣	One or more blanks must follow FRACHECK.
ENTITY=*entity addr*	*entity addr*: A-type address.
,CLASS=*'class name'* ,CLASS=*class name addr*	*class name*: 1- to 8-character class name *class name addr*: A-type address.
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER	**Default:** ATTR=READ
,ACEE=*acee addr*	*acee addr*: A-type address.
,WKAREA=*area addr*	*area addr*: A-type address.
,APPL=*'applname'* ,APPL=*applname addr*	*applname*: 1- to 8-character name *applname addr*: A-type address.
,INSTLN=*parm list addr*	*parm list addr*: A-type address.
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,MF=L	

The parameters are explained under the standard form of the FRACHECK macro instruction with the following exception:

,MF=L
specifies the list form of the FRACHECK macro instruction.

FRACHECK (Execute Form)

The execute form of the FRACHECK macro instruction is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede FRACHECK.
FRACHECK	
␣	One or more blanks must follow FRACHECK.
ENTITY=*entity addr*	*entity addr*: Rx-type address or register (2) - (12)
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,ATTR=*reg*	*reg*: Register (2) - (12)
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12)
,WKAREA=*area addr*	*area addr*: Rx-type address or register (2) - (12)
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12)
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12)
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6
,RELEASE=(,CHECK)	**Default:** RELEASE=1.6
,RELEASE=*(number,*CHECK)	
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) - (12)

The parameters are explained under the standard form of the FRACHECK macro instruction with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the FRACHECK macro instruction, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by the macro.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 5 on page 210.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the FRACHECK macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACDEF: Define a Resource to RACF

Note: On z/VM, you can use the RACDEF macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACDEF macro from a user's machine.

The RACDEF macro is used to define, modify, or delete resource profiles for RACF. It can also be used for special cases of authorization checking. RACF uses the resulting profiles to perform RACHECK authorization checking.

A RACF user can change or add the RACDEF parameters, OWNER, LEVEL, UACC, or AUDIT by means of the RACDEF preprocessing and postprocessing exit routines.

Note: Only callers in 24-bit addressing mode can issue this macro. Callers executing in 31-bit addressing mode, who want to use the RACDEF function, can code the RACROUTE macro.

RACDEF (Standard Form)

The standard form of the RACDEF macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACDEF	One or more blanks must precede RACDEF.
␣	One or more blanks must follow RACDEF.
ENTITY=*profile name addr*	*profile name addr*: A-type address, or register (2) - (12)
,VOLSER=*vol addr*	**Note:** VOLSER is required only for CLASS='DATASET' and DSTYPE not equal to M when a discrete profile name is used.
,TYPE=DEFINE	*new dsn addr*: A-type address, or register (2) - (12)
,TYPE=ADDVOL,OLDVOL= *old vol addr*	*old vol addr*: A-type address, or register (2) - (12)
,TYPE=CHGVOL,OLDVOL= *old vol addr*	
,TYPE=DELETE	**Default:** TYPE=DEFINE
,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	**Default:** DSTYPE=N

,INSTLN=*parm list addr*	*parm list addr*: A-type address, or register (2) - (12)
,CLASS='*class name*'	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address, or register (2) - (12) **Default:** CLASS='DATASET'
,MENTITY=*entity addr*	*entity addr*: A-type address, or register (2) - (12)
,MCLASS='*class name*'	*class name*: 1- to 8-character class name
,MCLASS=*class name addr*	**Default:** MCLASS='DATASET'
,MVOLSER=*volser addr*	*volser addr*: A-type address, or register (2) - (12)
,MGENER=ASIS	**Default:** MGENER=ASIS
,MGENER=YES	
,ACEE=*acee addr*	*acee addr*: A-type address, or register (2) - (12)
,UNIT=*unit addr*	*unit addr*: A-type address, or register (2) - (12)
,OWNER=*owner id addr*	*owner id addr*: A-type address, or register (2) - (12)
,LEVEL=*number*	**Default:** zero.
,LEVEL=*reg*	*reg*: Register (2) - (12)
,UACC=ALTER ,UACC=CONTROL ,UACC=UPDATE ,UACC=READ	
,UACC=NONE	
,UACC=*reg*	*reg*: Register (2) - (12)
,DATA=*data addr*	*data addr*: A-type address or register (2) - (12)
,AUDIT=NONE	**Note:** AUDIT is valid only if TYPE=DEFINE is specified.
,AUDIT=*audit value*	*audit value*: ALL, SUCCESS, or FAILURES

,AUDIT=(*audit value (access level),audit value(access level),...)*

access level: READ, UPDATE, CONTROL, or ALTER

Default: READ

,AUDIT=*(reg)*

reg: Register (2) - (12)

,RACFIND=YES

,RACFIND=NO

,CHKAUTH=YES

Default: CHKAUTH=NO

,CHKAUTH=NO

,GENERIC=YES

Default: GENERIC=ASIS

,GENERIC=ASIS

,WARNING=YES

Default: WARNING=NO

,WARNING=NO

Note: WARNING is valid only if TYPE=DEFINE is specified.

,RELEASE=*number*

number: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6

,FILESEQ=*reg*

reg: Register (2) - (12)

,FILESEQ=*number*

number: 1-9999

,EXPDT=*expir-date addr*

expir-date addr: A-type address or register (2) - (12)

,EXPDTX=*extended-expir-date addr*

extended-expir-date addr: A-type address or register (2) - (12)

,RETPD=*retn-period addr*

retn-period addr: A-type address or register (2) - (12)
Default: See description of parameter.

,ACCLVL=*(access value addr)*

access value addr: A-type address or register (2) - (12)

,ACCLVL=(*access value addr,parm list addr)*

parm list addr: A-type address, or register (2) - (12)

,TAPELBL=STD

Default: TAPELBL=STD

,TAPELBL=BLP

,TAPELBL=NL

,SECLVL=*addr*

addr: A-type address, or register (2) - (12)

,ERASE=YES ,ERASE=NO	**Default:** ERASE=NO
,NOTIFY=*notify-id addr*	*notify-id addr*: A-type address or register (2) - (12)
,ENVIR=VERIFY	Specifies that only verification is to be done. **Default:** Normal RACDEF processing.
,RESOWN=*resource owner addr*	A-type address, or register (2) - (12)
,STORCLA=*storage class addr*	A-type address, or register (2) - (12)
,MGMTCLA=*management type addr*	A-type address, or register (2) - (12)

The parameters are explained as follows:

ENTITY=*profile name addr*
specifies the address of the name of the discrete or generic profile that is to be defined to, modified, or deleted from RACF. The profile name is a 44-byte DASD data-set name for CLASS='DATASET' or a 6-byte volume-serial name for CLASS='DASDVOL' or CLASS='TAPEVOL'. The lengths of all other profile names are determined by the class-descriptor table. The name must be left-justified in the field and padded with blanks.

,VOLSER=*vol addr*
specifies the address of the volume-serial number:

- For TYPE=ADDVOL, of the new volume to be added to the definition of the data set.
- For TYPE=ADDVOL and CLASS='TAPEVOL', of the new volume being added to the tape-volume set identified by ENTITY.
- For TYPE=DEFINE and CLASS='DATASET', of the catalog (for a VSAM data set), or of the volume on which the data set resides (for a non-VSAM data set).

The volume serial number is optional if DSTYPE=M is specified; it is ignored if the profile name is generic.

The field pointed to by the specified address contains the volume serial number (padded to the right with blanks, if necessary, to make six characters).

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,TYPE=DEFINE
,TYPE=DEFINE,NEWNAME=*new dsn addr*
,TYPE=ADDVOL,OLDVOL=*old vol addr*
,TYPE=CHGVOL,OLDVOL=*old vol addr*
,TYPE=DELETE
specifies the type of action to be taken:

- TYPE=DEFINE. The definition of the resource is added to the RACF data set, and the current user is established as the owner of the defined entity.

- TYPE=DEFINE,NEWNAME=. If NEWNAME is specified, the address points to a 44-byte field containing the new name for the resource that is to be renamed.

  NEWNAME is valid with CLASS= DATASET, FILE, or DIRECTRY. NEWNAME is not valid with DSTYPE=T.
- TYPE=ADDVOL. The new volume is added to the definition of the specified resource. For the DATASET class, the OLDVOL address specifies a previous volume of a multivolume data set. For the TAPEVOL class, the ENTITY address specifies a previous volume of a tape volume set. This parameter applies only to discrete profiles.
- TYPE=CHGVOL. The volume serial number in the definition of the specified resource is changed from the old volume serial number identified in OLDVOL to the new volume serial number identified in the VOLSER parameter. This parameter applies only to discrete profiles. TYPE=CHGVOL is not valid with DSTYPE=T.
- TYPE=DELETE. The definition of the resource is removed from the RACF data set. (For a multivolume data set or a tape volume set, only the specified volume is removed from the definition.)

  If DSTYPE=T is specified, the data sets must be deleted in the reverse order they were created. For example, if file1 has dataset1, files2 has dataset2, and file3 has dataset3, you must do the RACDEF TYPE=DELETE,DSTYPE=T for file3, file2, and file1, in that order.

Note:

- If SETROPTS ADDCREATOR is in effect when a new DATASET or general resource profile is defined, the profile creator's user ID is placed on the profile access list with ALTER authority.
- If SETROPTS NOADDCREATOR is in effect when a new generic profile is defined, the profile creator's user ID is not placed on the profile's access list. If you use profile modeling when defining a generic profile, RACF copies the access list exactly. If the creator's user ID appeared in the model's access list, the same authority is copied to the new profile.
- If SETROPTS NOADDCREATOR is in effect when a new discrete DATASET or TAPEVOL profile is defined, the profile creator's user ID is placed on the profile's access list with ALTER authority. If you use profile modeling when defining one these profiles, if the creator's user ID appeared in the model's access list, the authority is created in the new profile with ALTER authority.
- If SETROPTS NOADDCREATOR is in effect when any other new discrete profile is defined, the profiles creator's user ID is not placed on the access list. If you use profile modeling when defining one of these profiles, RACF copies the access list exactly. If the creator's user ID appeared in the model's access list, the same authority is copied to the new profile.

,DSTYPE=N
,DSTYPE=V
,DSTYPE=M
,DSTYPE=T
specifies the type of data set associated with the request:

- N for non-VSAM
- V for VSAM
- M for model profile
- T for tape.

If DSTYPE=T is specified and tape data-set protection is not active, the processing is the same as for RACDEF CLASS='TAPEVOL'. Specify DSTYPE only for CLASS='DATASET'.

,INSTLN=*parm list addr*
specifies the address of an area that is to contain parameter information meaningful to the RACDEF installation exit routines. This information is passed to the installation exit routines when they are given control from the RACDEF routine.

The INSTLN parameter can be used by an application program acting as a resource manager that needs to pass information to the RACDEF installation exit routines.

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies that a profile is to be defined, modified, or deleted in the specified class. If an address is specified, the address must point to a 1-byte length field followed by the class name (for example, DATASET or TAPEVOL). The class name should be no longer than eight characters.

,MENTITY=*entity addr*
specifies the address of the name of the discrete or generic profile that is to be used as a model in defining the ENTITY profile. The profile can belong to any class, as specified by the MCLASS parameter, and can be either a discrete or a generic profile. MENTITY can be specified with TYPE=DEFINE but not with TYPE=DEFINE,NEWNAME=*new dsn addr*. The name is contained in a 44-byte field pointed to by the specified address. The name is left-justified in the field and padded with blanks.

,MCLASS=*'class name'*
,MCLASS=*class name addr*
specifies the class to which the profile defined by MENTITY= belongs. If an address is specified, the address must point to a 1-byte length field followed by the class name. The class name should be no longer than eight characters. The default is MCLASS='DATASET'.

,MVOLSER=*volser addr*
specifies the address of the volume serial number of the volume associated with the profile in the MENTITY operand. The field pointed to by the specified address contains the volume serial number, padded to the right with blanks if necessary to make six characters.

If you specify MENTITY and CLASS='DATASET', you must specify MVOLSER with the name of the volume serial number or with blanks.

If you specify it with blanks, the discrete MENTITY data-set profile name must be unique, meaning it has no duplicates on the database. In this case, RACF determines the correct MVOLSER.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,MGENER=ASIS
,MGENER=YES
specifies whether the profile name defined by MENTITY is to be treated as a generic name.

- If MGENER=ASIS is specified, the profile name is considered a generic only if it contains a generic character: an asterisk (*) or a percent sign (%).
- If MGENER=YES is specified, the profile name is considered a generic, even if it does not contain a generic character: an asterisk (*) or a percent sign (%).

MGENER is ignored if the GENCMD option on the RACF SETROPTS command is not specified for the class. (See *z/VM: RACF Security Server Command Language Reference*.)

,ACEE=*acee addr*
specifies the address of the ACEE to be used during RACDEF processing.

The ACEE should have been created as the result of a previous RACINIT invocation.

,UNIT=*unit addr*
specifies the address of a field containing unit information. If a unit address is specified, the unit information in the data-set profile is replaced by the unit information pointed to by this unit address. The unit address must point to a field containing a 1-byte length field (whose value can range from 4 through 8) followed by the actual unit information. If the value in the length field is 4, the unit information is assumed to contain a copy of the information in the UCBTYP field of the UCB. Otherwise the unit information is assumed to be in the generic form (for example, 3330-1).

UNIT is valid if TYPE=CHGVOL or TYPE=DEFINE is specified. It is ignored for generic names.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,OWNER =*owner id addr*
specifies the address of a field containing the profile owner's ID. OWNER is valid if TYPE=DEFINE is specified. The owner's ID must be a valid (RACF-defined) user ID or group name. The address must point to an 8-byte field containing the owner's name, left-justified and padded with blanks.

,LEVEL=*number*
,LEVEL=*reg*
specifies a level value for the profile. LEVEL is valid only if TYPE=DEFINE is specified. The level number must be a valid decimal number in the range 0 to 99. If a register is specified, its low-order byte must contain the binary representation of the number.

Note: RACF does not check the validity of this number if it has been added or modified by the RACDEF preprocessing or postprocessing exit routines.

,UACC=ALTER
,UACC=CONTROL
,UACC=UPDATE
,UACC=READ
,UACC=NONE
,UACC=*reg*
specifies a universal-access authority for the profile. UACC is valid only if TYPE=DEFINE is specified. UACC must contain a valid access authority (ALTER, CONTROL, UPDATE, READ, or NONE).

If a register is specified, the low-order byte must contain one of the following valid access authorities:

X'80' ALTER
X'40' CONTROL
X'20' UPDATE
X'10' READ
X'01' NONE

Note: RACF does not check the validity of the universal-access authority if it has been added or modified by the RACDEF preprocessing or postprocessing exit routine.

,DATA=*data addr*
specifies the address of a field that contains up to 255 characters of installation-defined data to be placed in the profile. DATA is valid only if TYPE=DEFINE is specified. The data address must point to a field containing a 1-byte length field (whose value can range from 0 to 255) followed by the actual installation-defined data.

,AUDIT=NONE
,AUDIT=*audit value*
,AUDIT=*(audit value(access level),audit value(access level),. . .)*
,AUDIT=*(reg)*
specifies the types of accesses and the access levels that are to be logged to the SMF data set. AUDIT is valid only if TYPE=DEFINE is specified.

For *audit value*, specify one of the following: ALL, SUCCESS, or FAILURES. You may, optionally, specify an *access level* (access authority) following each audit value.

Access Levels:

- READ: The default access level value, logs access attempts at any level.
- UPDATE: Logs access attempts at the UPDATE, CONTROL, and ALTER levels.
- CONTROL: Logs access attempts at the CONTROL and ALTER levels.
- ALTER: Logs access attempts at the ALTER level only.

Note: For more information about specific audit values and access levels, see *z/VM: RACF Security Server Command Language Reference*.

RACF resolves combinations of conflicting specifications by using the most encompassing specification. Thus, in the case of the following:

```
ALL(UPDATE),FAILURES(READ)
```

RACF assumes SUCCESS(UPDATE),FAILURES(READ).

For compatibility with previous releases, register notation can also be specified as AUDIT=*reg* if the register is not given the symbolic name ALL, SUCCESS, or FAILURES.

Logging is controlled separately for SUCCESS and FAILURES, and can also be suppressed or requested using the RACHECK postprocessing installation exit routine.

If a register is specified, its low-order byte must contain one of the following valid audit values:

Bit
Meaning

0
ALL

1
SUCCESS

2
FAILURES

3
NONE

4-5
Qualifier for SUCCESS

6-7
Qualifier for FAILURES

The qualifier codes are as follows:

00
READ

01
UPDATE

10
CONTROL

11
ALTER

Only one of bits 0 through 3 can be on. If ALL is specified, the two qualifier fields can be used to request different logging levels for successful and unsuccessful events.

,RACFIND=YES
,RACFIND=NO
specifies whether a discrete profile is involved in RACDEF processing. When TYPE=DEFINE is specified, RACFIND=YES means that a discrete profile is to be created. When TYPE=DELETE, DEFINE with NEWNAME, CHGVOL, or ADDVOL is specified, RACFIND=YES means that a discrete profile already exists.

RACFIND=NO means (when TYPE=DEFINE) that no discrete profile is to be created, but some authorization checking is required. For other types of action, no discrete profile should exist.

,CHKAUTH=YES
,CHKAUTH=NO
specifies whether or not RACF verifies that the user is authorized to perform the operation.

CHKAUTH=YES is valid when either TYPE=DEFINE,NEWNAME= or TYPE=DELETE is specified.

For DSTYPE=T, specifies that RACF verifies that the user is authorized to define a data set (TYPE=DEFINE), delete a data set (TYPE=DELETE), or add a volume (TYPE=ADDVOL).

,RELEASE=1.6|1.7|1.8|1.8.1
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 6 on page 224.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Execution-time validation of the compatibility between the list and execute forms of the RACDEF macro can be done by your specifying the CHECK subparameter on the execute form of the macro.

,FILESEQ=*number*
,FILESEQ=*reg*
specifies the file sequence number of a tape data set on a tape-volume or within a tape-volume set. The number must be in the range 1 through 9999. If a register is specified, it must contain the file-sequence number in the low-order halfword. If CLASS='DATASET' and DSTYPE=T are not specified, FILESEQ is ignored.

On z/VM, the z/VM operating system does not use RACDEF to provide security for tape volumes; however, you may have a tape-management product installed on z/VM that does use RACDEF.

,EXPDT=*expir-date addr*
,RETPD=*retn-period addr*
,EXPDTX=*extended expir-date addr*
specifies the address containing information about the data set's expiration date or RACF security retention period.

EXPDT=*expir-date addr* specifies the address of a 3-byte field containing the data set's expiration date. The date is given in packed decimal form as YYDDDF, where YY is the year and DDD is the day number. The year must be in the range 01 through 99, and the day number must be in the range 1 through 366. F allows the date to remain a positive integer when converted from packed decimal to hexadecimal. All fields are right-justified.

EXPDTX=*extended expir-date addr* specifies the address of a 4-byte field that contains the address of the data set's expiration date. The date is given in packed decimal form as CCYYDDDF, where CC is the century change greater than 19, YY is the year, and DDD is the day number. The year must be in the range 01 through 99. The day must be in the range 1 through 366. All fields are right-justified. When you want to represent 19 for the century, you must specify CC as 00; when you want to represent 20 for the century, you must specify CC as 01. F allows the date to remain a positive integer when converted from packed decimal to hexadecimal. To use this parameter, you must also specify RELEASE=1.8.

RETPD=*retn-period addr* specifies the address of a two-byte binary field containing the number of days after which RACF protection for the data set expires. The value specified must be in the range 1 through 65533. To indicate that there is no expiration date, specify 65534.

If you do not specify any of these parameters, a default RACF security retention period is obtained from the RETPD keyword specified on an earlier RACF SETROPTS command.

These parameters are valid only if CLASS='DATASET' and DSTYPE=T.

On z/VM, the z/VM operating system does not use RACDEF to provide security for tape volumes; however, you may have a tape-management product installed on z/VM that does use RACDEF.

,ACCLVL=*(access value addr)*
,ACCLVL=(*access value addr,parm list addr*)
specifies the tape-label access-level information for the z/OS tape-label functions. The address must point to a field containing a 1-byte length field (with a value that can range from 0 through 8) followed by an 8-character string that is passed to the RACDEF installation exit routines. The parameter-list

address points to a parameter list containing additional information to be passed to the RACDEF installation exit routines.

RACF does not check or modify this information.

TAPELBL=STD|BLP|NL
specifies the type of tape label processing to be done:

- STD: IBM or ANSI standard labels
- BLP: Bypass label processing
- NL: Unlabeled tapes.

For TAPELBL=BLP, the user must have the requested authority to the profile ICHBLP in the general-resource class FACILITY. For TAPELBL=NL or BLP, the user is not allowed to protect volumes with volume serial numbers in the format L*nnnnn*.

The TAPELBL parameter is passed to the RACDEF installation exits.

This parameter is primarily intended for use by data-management routines to indicate the label type from the LABEL keyword on the JCL statement.

This parameter is valid only for CLASS='DATASET' and DSTYPE=T, or CLASS='TAPEVOL'.

On z/VM, the z/VM operating system does not use RACDEF to provide security for tape volumes; however, you may have a tape-management product installed on z/VM that does use RACDEF.

,SECLVL=*addr*
specifies the address of a list of installation-defined security-level identifiers. Each identifier is a halfword, containing a value that corresponds to an installation-defined security-level name.

The identifiers must be in the range 1 through 254. Only one identifier may be passed in the list.

The list must start with a fullword containing the number of entries in the list (currently, only 0 or 1).

,ERASE=YES
,ERASE=NO
specifies whether the DASD data set, or the released space, is to be erased when it is deleted or part of its space is to be released for reuse.

- If ERASE=YES is specified, the data set is erased when it is deleted, or released for reuse.
- If ERASE=NO is specified, the data set is not be erased, deleted, or released.

Note: This parameter may be overridden by the RACF SETROPTS ERASE command.

The default is ERASE=NO.

Specify ERASE only for CLASS=DATASET.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,NOTIFY=*notify-id addr*
specifies the address of an 8-byte area containing the user ID of the RACF-defined user who is to be notified when an unauthorized attempt to access the resource protected by this profile is detected.

,GENERIC=YES
,GENERIC=ASIS
specifies whether the resource name is treated as a generic profile name. If GENERIC is specified with CLASS=DEFINE, NEWNAME, GENERIC applies to both the old and new names. GENERIC is ignored if the GENCMD option on the RACF SETROPTS command is not specified for the class. (See *z/VM: RACF Security Server Command Language Reference*.)

This keyword is designed primarily for use by RACF commands.

- If GENERIC=YES is specified, the resource name is considered a generic profile name, even if it does not contain a generic character: an asterisk (*) or a percent sign (%).

- If GENERIC=ASIS is specified, the resource name is considered a generic only if it contains a generic character: an asterisk (*) or a percent sign (%).

,WARNING=YES
,WARNING=NO
WARNING is valid only if TYPE=DEFINE is specified. If WARNING=YES is specified, access is granted to the resource and the event is logged as a warning if either the SUCCESS or FAILURES logging is requested.

,ENVIR=VERIFY
specifies that no profile is to be created, but that the user's authority to define or rename the resource or profile is to be checked, along with any other authorization processing that is necessary.

If you specify ENVIR, you must also specify RELEASE=1.8.1 or a later release number.

Note: If you do not specify ENVIR=VERIFY, normal RACDEF processing occurs.

,RESOWN=*resource owner address*
specifies the address of a field containing the resource owner's ID. If you specify RESOWN, you must also specify TYPE=DEFINE and the current RELEASE parameter. The resource owner's ID must be either a valid (RACF-defined) user ID or group name, or *NONE*. If the resource owner's ID is specified as *NONE*, RACF performs third-party RACHECK, using USERID=*NONE*. The address must point to a 2-byte field followed by the resource owner's name.

,MGMTCLA=*management class address*
specifies the address of a management class to which the resource owner must have authority. The address must point to an 8-byte field that contains a management-class name preceded by a halfword length. If you specify MGMTCLA, you must also specify TYPE=DEFINE, RESOWN, and RELEASE=1.8.1 or a later release number.

When MGMTCLA is specified, RACDEF processing invokes RACHECK processing to verify that the RESOWNER is authorized to the management class.

,STORCLA=*storage class address*
specifies the address of the storage class to which the resource owner must have authority. The address must point to a 2-byte field followed by the storage class name. If you specify STORCLA, you must also specify TYPE=DEFINE, RESOWN, and RELEASE=1.8.1 or a later release number.

When specified, RACDEF processing invokes RACHECK processing to verify that the RESOWNER is authorized to the storage class.

Parameters for RELEASE=1.6 through 1.8.1

The RELEASE values for which a parameter is valid are marked with an ‘X’.

Table 6. RACDEF Parameters for RELEASE= 1.6 through 1.8.1

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
ACEE=	X	X	X
ACCLVL=		X	X
AUDIT=	X	X	X
CHKAUTH=	X	X	X
CLASS=	X	X	X
DATA=	X	X	X
DSTYPE=N, V, or M	X	X	X
DSTYPE=T		X	X
ENTITY=	X	X	X

Table 6. RACDEF Parameters for RELEASE= 1.6 through 1.8.1 (continued)

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
ENVIR=			X
ERASE=		X	X
EXPDT=		X	X
EXPDTX=			X
FILESEQ=		X	X
GENERIC=	X	X	X
INSTLN=	X	X	X
LEVEL=	X	X	X
MCLASS=		X	X
MENTITY=	X	X	X
MGENER=		X	X
MGMTCLA=			X
MVOLSER=	X	X	X
NOTIFY=		X	X
OWNER=	X	X	X
RACFIND=	X	X	X
RELEASE=	X	X	X
RESOWN=			X
RETPD=		X	X
SECLVL=		X	X
STORCLA=			X
TAPELBL=		X	X
TYPE=	X	X	X
UACC=	X	X	X
UNIT=	X	X	X
VOLSER=	X	X	X
WARNING=	X	X	X

Return Codes and Reason Codes

If the return codes and reason codes you are receiving are not discussed in this macro, refer to the return codes and reason codes described with RACROUTE REQUEST=DEFINE in “Return Codes and Reason Codes” on page 70 for RACROUTE REQUEST=DEFINE (Standard Form).

When control is returned, register 15 contains one of the following return codes, and register 0 may contain a reason code.

Hexadecimal Code
Meaning

00
RACDEF has completed successfully. Register 0 contains one of the following reason codes:

00
Indicates a normal completion.

04
Indicates RACFIND=NO was specified and no generic profile applying to the data set was found.

04
RACDEF has completed processing. Register 0 contains one of the following reason codes:

00
Indicates the following:

- For TYPE=DEFINE, the resource name was previously defined.
- For TYPE=DEFINE,NEWNAME, the new resource name was previously defined.
- For TYPE=DELETE, the resource name was not previously defined.

04
Indicates for TYPE=DEFINE that the data-set name was previously defined on a different volume and that the option disallowing duplicate data sets was specified at IPL.

08
RACDEF has completed processing. Register 0 contains one of the following reason codes:

00
Indicates the following:

- For TYPE=DEFINE, the check for authority to allocate a data set or create a profile with the specified name has been failed.
- For TYPE=DELETE or TYPE=DEFINE,NEWNAME if CHKAUTH=YES is specified, the authorization check has been failed.
- For TYPE=ADDVOL,OLDVOL (or for TYPE=CHGVOL,OLDVOL) the old value was not defined.

04
Indicates for TYPE=DEFINE that no profile was found to protect the data set and that the RACF protect-all option is in effect.

08
Indicates TYPE=DEFINE (or TYPE=ADDVOL,OLDVOL or TYPE=CHGVOL,OLDVOL) and DSTYPE=T were specified, and the user is not authorized to define a data set on the specified volume.

0C
Indicates TYPE=DEFINE and DSTYPE=T were specified, and the user is not authorized to define a data set with the specified name.

10
Indicates DSTYPE=T or CLASS=TAPEVOL was specified, and the user is not authorized to specify LABEL=(,BLP).

20
Indicates the data-set owner is not authorized to use the specified DFP storage class.

24
Indicates the data-set owner is not authorized to use the specified DFP management class.

0C
For TYPE=DEFINE,NEWNAME, the old data-set name was not defined; or if the generation-data-group (GDG) modeling function is active, an attempt was made to rename a GDG name to a name that

requires the creation of a new profile; or if generic profile checking is active, the old data-set name was protected by a generic profile and there is no generic profile that will protect the new data-set name. This last case refers only to an attempt to rename an existing profile, which cannot be found.

10
For TYPE=DEFINE with MENTITY, the model resource was not defined.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACDEF macro; however, the list form of the macro does not have the proper RELEASE parameter. Macro processing terminates.

Example 1

Operation: Invoke RACF to define a discrete profile for a non-VSAM data-set residing on the volume pointed to by register 8. Register 7 points to the data set name. All successful requests for update authority to the data set are to be audited, as well as all unsuccessful ones.

```
RACDEF  ENTITY=(R7),VOLSER=(R8),CLASS='DATASET',
        AUDIT=(SUCCESS(UPDATE),FAILURES),
        RACFIND=YES
```

Example 2

Operation: Use the standard form of the RACDEF macro to define a discrete data-set profile for a non-VSAM DASD data set. The data set for which you are creating a profile is a non-VSAM DASD data set named DSNAME. It resides on a volume ID named VOLID. You want to create a discrete profile by specifying the RACFIND keyword. In addition, you want to notify the user called USERNAME of any access attempts that have been rejected because they exceed the UACC of READ that you are allowing.

```
RACDEF ENTITY=DSNAME,VOLSER=VOLID,CLASS='DATASET',UACC=READ,     X
       RACFIND=YES,NOTIFY=USERNAME,RELEASE=1.7
```

Example 3

Operation: Use the standard form of the macro to check the authority of a user to define a discrete data-set profile for a non-VSAM DASD data set, but do not actually build the profile. The name of the data set is DSNAME.

```
RACDEF ENTITY=DSNAME,VOLSER=VOLID,CLASS='DATASET',RACFIND=NO
```

Example 4

Operation: Use the standard form of the macro to define a generic data-set profile named PROFNAME. Use the discrete profile named MDELPROF whose volser is in MDELVOL as a model for the new profile. Notify the user named USERNAME of any access attempts that have been rejected because they exceed the UACC of READ which you are allowing.

```
RACDEF ENTITY=PROFNAME,CLASS='DATASET',GENERIC=YES,MENTITY=MDELPROF,  X
       MVOLSER=MDELVOL,UACC=READ,NOTIFY=USERNAME,RELEASE=1.7
```

Example 5

Operation: Use the standard form of the macro to define a tape-volume profile for a volume whose ID is VOLID. Allow a universal-access level of READ.

```
RACDEF ENTITY=VOLID,CLASS='TAPEVOL',UACC=READ
```

Example 6

Operation: Use the standard form of the macro to delete a discrete data-set profile named DSNAME, located on the volume named VOLID.

```
RACDEF TYPE=DELETE,ENTITY=DSNAME,VOLSER=VOLID,CLASS='DATASET'
```

RACDEF (List Form)

The list form of the RACDEF macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACDEF	One or more blanks must precede RACDEF.
␣	One or more blanks must follow RACDEF.
ENTITY=*profile name addr*	**Note:** ENTITY must be specified on either the list or the execute form of the macro.
,VOLSER=*vol addr*	**Note:** VOLSER is required (on either LIST or EXECUTE) only for CLASS='DATASET' and DSTYPE not equal to M when a discrete profile name is used.
,TYPE=DEFINE	
,TYPE=DEFINE,NEWNAME= *new dsn addr*	*new dsn addr*: A-type address
,TYPE=ADDVOL,OLDVOL= *old vol addr*	*old vol addr*: A-type address
,TYPE=CHGVOL,OLDVOL= *old vol addr*	
,TYPE=DELETE	**Default:** TYPE=DEFINE

,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	**Default:** DSTYPE=N
,INSTLN=*parm list addr*	*parm list addr*: A-type address
,CLASS=*'class name'* ,CLASS=*class name addr*	*class name*: 1- to 8-character class name. **Default:** CLASS='DATASET'
,MENTITY=*entity addr*	*entity addr*: A-type address
,MCLASS=*'class name'* ,MCLASS=*class name addr*	*class name*: 1- to 8-character class name **Default:** MCLASS='DATASET'
,MVOLSER=*volser addr*	*volser addr*: A-type address
,MGENER=ASIS ,MGENER=YES	**Default:** MGENER=ASIS
,ACEE=*acee addr*	*acee addr*: A-type address
,UNIT=*unit addr*	*unit addr*: A-type address
,OWNER=*owner id addr*	*owner id addr*: A-type address
,LEVEL=*number*	**Default:** Zero.
,UACC=ALTER ,UACC=CONTROL ,UACC=UPDATE ,UACC=READ ,UACC=NONE	
,DATA=*data addr*	*data addr*: A-type address
,AUDIT=NONE ,AUDIT=*audit value* ,AUDIT=(*audit value (access level),audit value(access level))*	*audit value*: ALL, SUCCESS, or FAILURES **Default:** READ

,RACFIND=YES
,RACFIND=NO

,CHKAUTH=YES
,CHKAUTH=NO

Default: CHKAUTH=NO

,GENERIC=YES
,GENERIC=ASIS

Default: GENERIC=ASIS

,WARNING=YES
,WARNING=NO

Default: WARNING=NO

Note: Warning is valid only if TYPE=DEFINE is specified.

,RELEASE=*number*

Default: RELEASE=1.6

,FILESEQ=*number*

number: 1-9999

,EXPDT=*expir-date addr*

expir-date addr: A-type address

,RETPD=*retn-period addr*

retn-period addr: A-type address

,EXPDTX=*ex-expir-date addr*

extended expir-date addr: A-type address

,ENVIR=VERIFY

Default: Normal RACDEF processing.

,RESOWN=*resource owner addr*

resource owner addr: A-type address

,STORCLA=*storage class addr*

storage class addr: A-type address

,MGMTCLA=*management type*

Default: See description of parameter.

,ACCLVL=*(access value addr)*

access value addr: A-type address

,ACCLVL=(*access value addr,parm list addr)*

parm list addr: A-type address

,TAPELBL=STD
,TAPELBL=BLP
,TAPELBL=NL

Default: TAPELBL=STD

,SECLVL=*addr*

addr: A-type address

,ERASE=YES ,ERASE=NO	**Default:** ERASE=NO
,NOTIFY=*notify-id addr*	*notify-id addr*: A-type address
,MF=L	

The parameters are explained under the standard form of the RACDEF macro instruction with the following exception:

,MF=L
specifies the list form of the RACDEF macro instruction.

RACDEF (Execute Form)

The execute form of the RACDEF macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACDEF	One or more blanks must precede RACDEF.
␣	One or more blanks must follow RACDEF.
ENTITY=*profile name addr*	**Note:** ENTITY must be specified on either the list or the execute form of the macro.
,VOLSER=*vol addr*	**Note:** VOLSER is required only for CLASS='DATASET' and DSTYPE not equal to M when a discrete profile name is used.
,TYPE=DEFINE ,TYPE=DEFINE,NEWNAME=*new dsn addr*	*new dsn addr*: Rx-type address or register (2) - (12)
,TYPE=ADDVOL,OLDVOL=*old vol addr* ,TYPE=CHGVOL,OLDVOL=*old vol addr* ,TYPE=DELETE	*old vol addr*: Rx-type address or register (2) - (12)

,DSTYPE=N
,DSTYPE=V
,DSTYPE=M
,DSTYPE=T

,INSTLN=*parm list addr* — *parm list addr*: Rx-type address or register (2) - (12)

,CLASS=*class name addr* — *class name addr*: Rx-type address or register (2) - (12)

,MENTITY=*entity addr* — *entity addr*: Rx-type address or register (2) - (12)

,MCLASS=*class name addr* — *class name addr*: Rx-type address or register (2) - (12)

,MVOLSER=*volser addr* — *volser addr*: Rx-type address or register (2) - (12)

,MGENER=ASIS
,MGENER=YES

,ACEE=*acee addr* — *acee addr*: Rx-type address or register (2) - (12)

,UNIT=*unit addr* — *unit addr*: Rx-type address or register (2) - (12)

,OWNER=*owner id addr* — *owner id addr*: Rx-type address or register (2) - (12)

,LEVEL=*number*
,LEVEL=*reg* — *reg*: Register (2) - (12)

,UACC=ALTER
,UACC=CONTROL
,UACC=UPDATE
,UACC=READ
,UACC=NONE
,UACC=*reg* — *reg*: Register (2) - (12)

,DATA=*data addr* — *data addr*: Rx-type address or register (2) - (12)

,AUDIT=NONE
,AUDIT=*audit value* — *audit value*: ALL, SUCCESS, or FAILURES

,AUDIT=(*audit value (access level),audit value(access level)*)	*access level*: READ, UPDATE, CONTROL, or ALTER
,AUDIT=*(reg)*	*reg*: Register (2) - (12)
,RACFIND=YES ,RACFIND=NO	
,CHKAUTH=YES ,CHKAUTH=NO	
,GENERIC=YES ,GENERIC=ASIS	
,WARNING=YES ,WARNING=NO	**Note:** Warning is valid only if TYPE=DEFINE is specified.
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=*(number,*CHECK)	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,FILESEQ=*reg* ,FILESEQ=*number*	*reg*: Register (2) - (12) *number*: 1-9999
,EXPDT=*expir-date addr* ,RETPD=*retn-period addr* ,EXPDTX=*extended-expir-date addr*	*expir-date addr*: Rx-type address or register (2) - (12) *retn-period addr*: Rx-type address or register (2) - (12) *extended expir-date addr*: Rx-type address or register (2) - (12)
,ENVIR=VERIFY	Specifies that only verification is to be done.
,RESOWN=*resource owner addr*	*resource owner addr*: Rx-type address or register (2) - (12)
,STORCLA=*storage class addr*	*storage class addr*: Rx-type address or register (2) - (12)
,MGMTCLA=*management class addr*	*management class addr*: Rx-type address or register (2) - (12)
,ACCLVL=*(access value addr)*	*access value addr*: Rx-type address or register (2) - (12).

,ACCLVL=(*access value addr,parm list addr)*	*parm list addr*: Rx-type address or register (2) - (12)
,TAPELBL=STD ,TAPELBL=BLP ,TAPELBL=NL	
,SECLVL=*addr*	*addr*: Rx-type address or register (2) - (12)
,ERASE=YES ,ERASE=NO	
,NOTIFY=*notify-id addr*	*notify-id addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) or (2) - (12)

The parameters are explained under the standard form of the RACDEF macro instruction with the following exceptions:

,MF=(E,*ctrl addr)*
specifies the execute form of the RACDEF macro, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 6 on page 224.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACDEF macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACHECK: Check RACF Authorization

The RACHECK macro is used to provide authorization checking when a user requests access to a RACF-protected resource.

Note:

1. Only callers in 24-bit addressing mode can issue this macro. Callers executing in 31-bit addressing mode who want to use the RACHECK function can code the RACROUTE macro.

2. **On z/VM,** you can use the RACHECK macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACHECK macro from a user's machine.

RACHECK (Standard Form)

The standard form of the RACHECK macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACHECK	One or more blanks must precede RACHECK.
␣	One or more blanks must follow RACHECK.
PROFILE=*profile addr*	*profile addr*: A-type address or register (2) - (12)
ENTITY=(*resource name addr*)	*resource name addr*: A-type address or register (2) - (12).
ENTITY=(*resource name addr*,CSA)	
ENTITY=(*resource name addr*,PRIVATE)	
ENTITY=(*resource name addr*,NONE)	
,VOLSER=*vol addr*	**Note:** VOLSER is required only for CLASS='DATASET' and DSTYPE not equal to M when a discrete profile name is used and only when ENTITY is also coded.
,CLASS='*class name*'	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,RELEASE=*number*	**Default:** RELEASE=1.6
,ATTR=READ	*reg*: Register (2) - (12)
,ATTR=UPDATE	**Default:** ATTR=READ
,ATTR=CONTROL ,ATTR=ALTER ,ATTR=*reg*	

,DSTYPE=N
,DSTYPE=V
,DSTYPE=M
,DSTYPE=T

Default: DSTYPE=N

,INSTLN=*parm list addr*

parm list addr: A-type address or register (2) - (12).

,LOG=ASIS
,LOG=NOFAIL
,LOG=NONE
,LOG=NOSTAT

Default: LOG=ASIS

,OLDVOL=*old vol addr*

old vol addr: A-type address or register (2) - (12).

,APPL=*'applname'*

applname: 1- to 8-character name

,APPL=*applname addr*

A-type address or register (2) - (12).

,ACEE=*acee addr*

acee addr: A-type address or register (2) - (12).

,ACCLVL=(*access value addr)*

access value addr: A-type address or register (2) - (12).

,ACCLVL=(*access value addr,parm list addr)*

parm list addr: A-type address or register (2) - (12).

,RACFIND=YES
,RACFIND=NO

,GENERIC=YES
,GENERIC=ASIS

Default: GENERIC=ASIS

,FILESEQ=*reg*

reg: Register (2) - (12).

,FILESEQ=*number*

number: 1-9999

,TAPELBL=STD
,TAPELBL=BLP
,TAPELBL=NL

Default: TAPELBL=STD

,STATUS=NONE
,STATUS=ERASE

Default: STATUS=NONE

,USERID=*'userid'*	*userid*: 1- to 8-character user ID
,USERID=*userid addr*	*userid addr*: A-type address or register (2) - (12)
,GROUPID=*'groupid'*	*groupid*: 1- to 8-character group ID
,GROUPID=*groupid addr*	*groupid addr*: A-type address or register (2) - (12)

The parameters are explained as follows:

,PROFILE=*profile addr*
,ENTITY=(*resource name addr*)
,ENTITY=(*resource name addr*, CSA)
,ENTITY=(*resource name addr*,PRIVATE)
,ENTITY=(*resource name addr*,NONE)

PROFILE=*profile addr* specifies that RACF authorization checking is to be performed for the resource whose profile is pointed to by the specified address. This profile must be supplied by ENTITY=(*xxx*,CSA). A profile supplied by RACLIST is not acceptable.

For the ENTITY keyword, the resource name is a 44-byte DASD data-set name for CLASS='DATASET', or a 6-byte volume serial number for CLASS='DASDVOL' or CLASS='TAPEVOL'. The length of all other resource names is determined from the class-descriptor tables.

- ENTITY=(*resource name addr*) specifies that RACF authorization checking is to be performed for the resource whose name is pointed to by the specified address. The name must be left-justified in the field and padded with blanks.
- ENTITY=(*resource name addr*,CSA) specifies that RACF authorization checking is to be performed for the indicated resource, and that a copy of the profile is to be maintained in main storage. The storage acquired for the profile is obtained from the common storage area (CSA), and is fetch-protected, key 0 storage. The issuer of RACHECK must free this storage when the profile is no longer needed. (The profile subpool number and length are part of the profile data returned.) If CSA is specified and the return code produced by the RACHECK macro instruction is 00 or 08, the address of the profile is returned in register 1.

  By establishing and maintaining a resource profile, the resource manager can reduce the I/O required to perform RACF authorization checks on frequently accessed resources.
- ENTITY=(*resource name addr*,PRIVATE) PRIVATE specifies the same as CSA except that RACHECK returns the profile in the user's private area rather than in common storage, and the name field contains the name of the returned profile instead of the name of the resource that was specified on the ENTITY keyword. The issuer of RACHECK must free this storage when the profile is no longer needed. (The profile subpool number and length are returned as well as the profile data.)
- ENTITY=(*resource name addr*,NONE) specifies the same as ENTITY=resource name address. However, no profile is returned.

,VOLSER=*vol addr*

specifies the volume serial number, as follows:

- For VSAM DASD data sets, this is the volume serial number of the catalog controlling the data set.
- For non-VSAM DASD data sets and tape data sets, this is the volume serial number of the volume on which the data set resides.

The volume serial number is optional if DSTYPE=M is specified; it is ignored if the profile name is generic.

The field pointed to by the specified address contains the volume serial number, padded to the right with blanks if necessary to make six characters. VOLSER= is only valid and must be supplied with CLASS='DATASET', (unless DSTYPE=M is specified) and if ENTITY is also coded.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,CLASS=*'class name'*
,CLASS=*class name addr*
specifies that RACF authorization checking is to be performed for a resource of the specified class. If an address is specified, the address must point to a 1-byte field indicating the length of the class name, followed by the class name.

,RELEASE=1.6|1.7|1.8|1.8.1|1.8.2
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For instance, to use the STATUS parameter, you must be using RACF 1.7 or later on your system and specify RELEASE=1.7 or later. For the parameters that are valid for RELEASE=1.6 and later, see Table 8 on page 242.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time.

,ATTR=READ
,ATTR=UPDATE
,ATTR=CONTROL
,ATTR=ALTER
,ATTR=*reg*
specifies the access authority of the user or group permitted access to the resource for which RACF authorization checking is to be performed:

READ: RACF user or group can open the resource only to read.
UPDATE: RACF user or group can open the resource to write or read.
CONTROL: For VSAM data sets, RACF user or group has authority equivalent to the VSAM control password. For non-VSAM data sets and other resources, RACF user or group has UPDATE authority.
ALTER: RACF user or group has total control over the resource.

If a register is specified, the register must contain one of the following codes in the low-order byte of the register:

X'02' READ
X'04' UPDATE
X'08' CONTROL
X'80' ALTER.

,DSTYPE=N
,DSTYPE=V
,DSTYPE=M
,DSTYPE=T
specifies the type of data set associated with the request:

- N for non-VSAM
- V for VSAM
- M for model profile
- T for tape.

If DSTYPE=T is specified and tape data-set protection is not active, the processing is the same as for RACHECK CLASS='TAPEVOL'.

DSTYPE should be specified only for CLASS='DATASET'.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,INSTLN=*parm list addr*
specifies the address of an area that is to contain parameter information meaningful to the RACHECK installation exit routine. This information is passed to the installation exit routine when it is given control by RACHECK.

The INSTLN parameter can be used by an application program acting as a resource manager that needs to pass information to the RACHECK installation exit routine.

,LOG=ASIS
,LOG=NOFAIL
,LOG=NONE
,LOG=NOSTAT
specifies the types of access attempts to be recorded in the SMF data file:

ASIS: RACF records the event in the manner specified in the profile that protects the resource.
NOFAIL: If the authorization check fails, the attempt is not recorded. If the authorization check succeeds, the attempt is recorded as in ASIS.
NONE: The attempt is not to be recorded.
NOSTAT: The attempt is not to be recorded. No logging is to occur, and no resource statistics are to be updated (including messages and SMF records).

,OLDVOL=*old vol addr*
specifies a volume serial:

- For CLASS='DATASET', within the same multivolume data set specified by VOLSER=.
- For CLASS='TAPEVOL', within the same tape volume specified by ENTITY=.

RACF authorization checking verifies that the OLDVOL specified is part of the same multivolume data set or tape-volume set.

The specified address points to the field that contains the volume serial number, padded to the right with blanks if necessary to make six characters.

On z/VM, data sets may exist on OS or DOS minidisks or on tape volumes, but the z/VM operating system does not use RACROUTE to provide security for these data sets.

,APPL='*applname*'
,APPL=*applname addr*
specifies the name of the application requesting authorization checking. The *applname* is not used for the authorization-checking process but is made available to the installation exit routine or routines called by the RACHECK routine. If the address is specified, the address must point to an 8-byte field containing the application name, left-justified and padded with blanks.

,ACEE=*acee addr*
specifies the address of the ACEE to be used during RACHECK processing. If no ACEE is specified, RACF uses the TASK ACEE pointer (TCBSENV) in the extended TCB. If the TASK ACEE pointer is zero, RACF uses the main ACEE for the address space. The main ACEE is pointed to by the ASXBSENV field of the address-space extension block.

The ACEE used should have been created as the result of a previous RACINIT invocation.

,ACCLVL=(*access value addr*)
,ACCLVL=(*access value addr,parm list addr*)
specifies the tape-label access-level information for the z/OS tape-label functions. The access value pointed to by the specified address is a 1-byte length field, containing the value (0-8) of the length of the following data, followed by an 8-character string that is passed to the RACHECK installation exit routines. The optional parameter list pointed to by the specified address contains additional information to be passed to the RACHECK installation exit routines. RACF does not inspect or modify this information.

,RACFIND=YES
,RACFIND=NO
indicates whether or not the resource is protected by a discrete profile. The RACF processing and the possible return codes are given in Table 7 on page 240.

Note: In all cases, a return code of X'0C' is also possible if the OLDVOL specified was not part of the multivolume data set defined by VOLSER, or it was not part of the same tape volume defined by ENTITY.

,GENERIC=YES
,GENERIC=ASIS
specifies whether the resource name is to be treated as a generic profile name. If GENERIC is specified with CLASS=DEFINE, NEWNAME, then GENERIC applies to both the old and new names. GENERIC is ignored if the GENCMD option on the RACF SETROPTS command is not specified for the class. (See *z/VM: RACF Security Server Command Language Reference*.)

This keyword is designed primarily for use by RACF commands.

- If GENERIC=YES is specified, the resource name is considered a generic profile name, even if it does not contain either of the generic characters: an asterisk (*) or a percent sign (%).
- If GENERIC=ASIS is specified, the resource name is considered a generic only if it contains either of the generic characters: an asterisk (*) or a percent sign (%).

Table 7. Types of Profile Checking Performed by RACHECK

Operand	Generic Profile Checking Inactive	Generic Profile Checking Active
RACFIND=YES	Look for discrete profile; if found, exit with return code 00 or 08. If no discrete profile is found, exit with return code 08.	Look for discrete profile; if found, exit with return code 00 or 08. Look for generic profile; if found, exit with return code 00 or 08. Exit with return code 08 if neither a discrete nor a generic profile is found.
RACFIND=NO	No checking. Exit with return code 04.	Look for generic profile; if found, exit with return code 00 or 08. If not found, exit with return code 04.
RACFIND not specified	Look for discrete profile; if found, exit with return code 00 or 08. If no discrete profile is found, exit with return code 04.	Look for discrete profile; if found, exit with return code 00 or 08. Look for generic profile; if found, exit with return code 00 or 08. Exit with return code 04 if neither a discrete nor a generic profile is found.

,FILESEQ=*number*
,FILESEQ=*reg*
specifies the file-sequence number of a tape data set on a tape volume or within a tape-volume set. The value must be in the range 1 through 9999. If a register is specified, it must contain the file sequence number in the low-order halfword. If CLASS='DATASET' and DSTYPE=T are not specified, FILESEQ is ignored.

On z/VM, the z/VM operating system does not use RACDEF to provide security for tape volumes; however, you may have a tape-management product installed on z/VM that does use RACDEF.

,TAPELBL=STD|BLP|NL
specifies the type of tape-label processing to be done:

- STD: IBM or ANSI standard labels
- BLP: Bypass label processing
- NL: Unlabelled tapes.

For TAPELBL=BLP, the user must have the requested authority to the profile ICHBLP in the general-resource class FACILITY. For TAPELBL=NL or BLP, the user is not allowed to protect volumes with volume serial numbers in the format "L*nnnnn*".

This parameter is primarily intended for use by data-management routines to indicate the label type from the LABEL keyword on the JCL statement.

This parameter is valid only for CLASS='DATASET' and DSTYPE=T, or CLASS='TAPEVOL'.

On z/VM, the z/VM operating system does not use RACDEF to provide security for tape volumes; however, you may have a tape-management product installed on z/VM that does use RACDEF.

,STATUS=NONE|ERASE
specifies whether or not RACHECK is to return the erase status of the given data set. This parameter is valid only for CLASS='DATASET' and a DSTYPE value other than T.

,USERID='*user ID*'
,USERID=*user ID addr*
specifies the user ID that RACF uses to perform third-party RACHECK. This is an 8-character field that is left-justified and padded to the right with blanks.

If USERID is specified when the caller invokes RACHECK, RACF verifies that user's authority to the given entity; RACF disregards the user ID associated with the ACEE of the caller.

For third-party RACHECK, RACF performs the following steps:

1. Checks to see whether the USERID keyword is *NONE* and GROUPID is not specified. If so, then RACF creates a default user (null) ACEE, which it uses to perform the RACHECK.
2. If not, checks to see whether an additional (third-party) ACEE already exists, chained off the current caller's ACEE or the ACEE specified in the ACEE= keyword.
3. If so, checks to see whether the user ID in that ACEE matches the one specified on the USERID keyword. If so, RACHECK uses the existing ACEE and avoids RACINIT processing.
4. If USERID is specified and RACHECK does not find an additional (third-party) ACEE, or the user ID in the ACEE does not match the user ID specified on the USERID keyword, then RACHECK creates a third-party ACEE based on the USERID keyword.
5. If the GROUPID keyword is specified in addition to the USERID keyword, and a third-party ACEE already exists, the group ID of the existing third-party ACEE must also match the group ID specified on the GROUPID keyword. If the GROUPID keywords do not match, RACHECK creates a third-party ACEE based on the USERID keyword.

Note: If the calling program does not specify the GROUPID keyword, the internal RACINIT function uses the default group associated with the specified user ID.

Note: If the user ID is *NONE* and a GROUPID has not been specified, then a default user (null) ACEE is created and used to satisfy RACHECK processing.

,GROUPID='*groupid*'
,GROUPID=*groupid addr*
specifies the group ID that RACF uses to perform third-party RACHECK.

If the calling program wants a third-party RACHECK performed on the group ID rather than the user ID, the USERID keyword must be specified as *NONE*. When the caller invokes third-party RACHECK, RACF verifies the authority of the group ID to the requested resource; RACF disregards the group ID associated with the ACEE of the caller. For third-party RACHECK, RACF performs the following steps:

- Checks to see whether an additional (third-party) ACEE already exists, chained off the caller's ACEE, or the ACEE specified in the ACEE= keyword.
- If so, checks to see whether the group ID matches that specified on the GROUPID keyword. If so, RACHECK uses that ACEE and avoids RACINIT processing.
- If GROUPID is specified and RACHECK does not find an additional (third-party) ACEE, or the group ID in the ACEE does not match the group ID specified on the GROUPID keyword, RACHECK creates a third-party ACEE based on the GROUPID keyword.

Parameters for RELEASE=1.6 through 1.8.2

The RELEASE values for which a specific parameter is valid are marked with an X.

Table 8. RACHECK Parameters for RELEASE=1.6 through 1.8.2

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8, 1.8.1, or 1.8.2
ACEE=	X	X	X
ACCLVL=	X	X	X
APPL=	X	X	X
ATTR=	X	X	X
CLASS=	X	X	X
DSTYPE=N, V, or M	X	X	X
DSTYPE=T		X	X
ENTITY=	X	X	X
FILESEQ=		X	X
GENERIC=	X	X	X
GROUPID=			X
INSTLN=	X	X	X
LOG=	X	X	X
OLDVOL=	X	X	X
OWNER=	X	X	X
PROFILE=	X	X	X
RACFIND=	X	X	X
RELEASE=	X	X	X
STATUS=		X	X
TAPELBL=		X	X
USERID=			X
VOLSER=	X	X	X

Return Codes and Reason Codes

If the return codes and reason codes you are receiving are not discussed in this macro, refer to the return codes and reason codes described with RACROUTE REQUEST=AUTH on page "RACROUTE REQUEST=AUTH (Standard Form)" on page 35.

When control is returned, register 15 contains one of the following return codes, and register 0 may contain a reason code.

Hexadecimal Code
Meaning

00
The user is authorized by RACF to obtain use of a RACF-protected resource. Register 0 contains one of the following reason codes:

00
Indicates a normal completion.

04
Indicates STATUS=ERASE was specified and the data set is to be erased when scratched.

Otherwise, indicates that the warning status of the resource was requested by the RACHECK issuer setting bit 10 at offset 12 decimal in the RACHECK parameter list and the resource is in warning mode.

10
When CLASS=TAPEVOL, indicates the TAPEVOL profile contains a TVTOC.

20
When CLASS=TAPEVOL, indicates that the TAPEVOL profile can contain a TVTOC, but currently does not (for a scratch pool volume).

24
When CLASS=TAPEVOL, indicates that the TAPEVOL profile does not contain a TVTOC.

04
The specified resource is not protected by RACF. Register 0 contains the following reason code:

00
Indicates a normal completion.

08
The user is **not** authorized by RACF to obtain use of the specified RACF-protected resource. Register 0 contains the following reason code:

00
Indicates a normal completion.

04
Indicates STATUS=ERASE was specified and the data set is to be erased when scratched.

08
Indicates DSTYPE=T or CLASS='TAPEVOL' was specified and the user is not authorized to use the specified volume.

0C
Indicates the user is not authorized to use the data set.

10
Indicates DSTYPE=T or CLASS='TAPEVOL' was specified and the user is not authorized to specify LABEL=(,BLP).

0C
The OLDVOL specified was not part of the multivolume data set defined by VOLSER, or it was not part of the same tape volume defined by ENTITY.

10
RACINIT issued by third-party RACHECK failed. Register 0 contains the RACINIT return code.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACHECK macro; however, the list form of the macro does not have the proper RELEASE parameter. Macro processing terminates.

Example 1

Operation: Perform RACF authorization checking using the standard form, for a non-VSAM data set residing on the volume pointed to by register 8. Register 7 points to the data-set name, and the RACF user is requesting the highest level of control over the data set. The "RACF-indicated" bit in the data set's DSCB is on. Logging and statistics updates are **not** to be done.

```
RACHECK  ENTITY=((R7)),VOLSER=(R8),CLASS='DATASET',              X
         ATTR=ALTER,RACFIND=YES,LOG=NOSTAT
```

Example 2

Operation: Perform RACF authorization checking using the standard form, for a non-VSAM data set controlled by the catalog pointed to by register 8. Register 7 points to the data-set name, and the RACF user is requesting the highest level of control over the data set. The "RACF-indicated" bit in the data set's DSCB is on.

```
RACHECK ENTITY=((R7)),VOLSER=(R8),CLASS='DATASET',              X
               ATTR=ALTER,RACFIND=YES
```

Example 3

Operation: Perform RACF authorization checking using the standard form, for a VSAM data set residing on the volume pointed to by register 8. Register 7 points to the data-set name, and the RACF user is requesting the data set for Read only. Register 4 points to an area containing additional parameter information.

```
RACHECK ENTITY=((R7)),VOLSER=(R8),CLASS='DATASET',              X
               DSTYPE=V,INSTLN=(R4)
```

Example 4

Operation: Using the standard form, perform RACF authorization checking for a tape volume for Read access only. The tape volume is pointed to by register 8 and the volume's access level is in register 5.

```
RACHECK ENTITY=((R8)),CLASS='TAPEVOL',ATTR=READ,                X
               ACCLVL=((R5))
```

Example 5

Operation: Using the standard form, perform third party RACF authorization checking for a data set for Read access only for a user. Register 7 points to the data-set name, and register 8 points to the volume on which the data set resides.

```
RACHECK ENTITY=((R7)),VOLSER=(R8),CLASS='DATASET',              X
               ATTR=READ,USERID='SOMEUSER'
```

Example 6

Operation: Using the standard form, perform third-party RACF authorization checking for a data set for Read access only for a group. Register 7 points to the data-set name, and register 8 points to the volume on which the data set resides.

```
RACHECK ENTITY=((R7)),VOLSER=(R8), CLASS='DATASET',ATTR=READ     X
               USERID='*NONE*',GROUPID='ANYGROUP'
```

Example 7

Operation: Using the standard form, perform third-party RACF authorization checking for a data set for a user connected to a group. Register 7 points to the data-set name, and register 8 points to the volume on which the data set resides.

```
RACHECK ENTITY=((R7)),VOLSER=(R8),CLASS='DATASET',ATTR=READ      X
               USERID='SOMEUSER',GROUPID='ANYGROUP'!
```

Example 8

Operation: Using the standard form, perform third-party RACF authorization checking for a data set (with Read-only access). Register 7 points to the data-set name, and register 8 points to the volume on which the data set resides. A is an 8-byte declared field padded with zeros.

```
RACHECK ENTITY=((R7)), VOLSER=(R8),CLASS='DATASET',              X
               ATTR=READ,USERID=A
```

RACHECK (List Form)

The list form of the RACHECK macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACHECK	One or more blanks must precede RACHECK.
␣	One or more blanks must follow RACHECK.
ENTITY=(*resource name addr*,NONE)	**Note:** PROFILE or ENTITY is required on either the list or the execute form of the macro.
,VOLSER=*vol addr*	**Note:** VOLSER is required on either the list or the execute form of the macro, but only for CLASS='DATASET' and DSTYPE not equal to M when a discrete profile name is used. If required, VOLSER must be specified on either the list or the execute form of the macro.
,CLASS='*class name*' ,CLASS=*class name addr*	*class name*: 1- to 8-character name *class name addr*: A-type address
,RELEASE=*number*	**Default:** RELEASE=1.6
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER	**Default:** ,ATTR=READ
,DSTYPE=N ,DSTYPE=V	**Default:** DSTYPE=N

,DSTYPE=M	
,DSTYPE=T	
,INSTLN=*parm list addr*	*parm list addr*: A-type address.
,LOG=ASIS	**Default:** LOG=ASIS
,LOG=NOFAIL	
,LOG=NONE	
,OLDVOL=*old vol addr*	*old vol addr*: A-type address.
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address.
,ACEE=*acee addr*	*acee addr*: A-type address.
,ACCLVL=(*access value addr)*	*access value addr*: A-type address
,ACCLVL=(*access value addr,parm list addr)*	*parm list addr*: A-type address
,RACFIND=YES	
,RACFIND=NO	
,GENERIC=YES	**Default:** GENERIC=ASIS
,GENERIC=ASIS	
,FILESEQ=*number*	*number*: 1-9999
,TAPELBL=STD	**Default:** TAPELBL=STD
,TAPELBL=BLP	
,TAPELBL=NL	
,STATUS=NONE	**Default:** STATUS=NONE
,STATUS=ERASE	
,USERID=*'userid'*	*userid*: 1- to 8- character user ID
,USERID=*userid addr*	*userid addr*: A-type address
,GROUPID=*'groupid'*	*groupid*: 1- to 8- character group ID

,GROUPID=*groupid addr*

groupid addr: A-type address

,MF=L

The parameters are explained under the standard form of the RACHECK macro with the following exception:

,MF=L
specifies the list form of the RACHECK macro instruction.

RACHECK (Execute Form)

The execute form of the RACHECK macro is written as follows:

name

name: Symbol. Begin *name* in column 1.

␣
RACHECK

One or more blanks must precede RACHECK.

␣

One or more blanks must follow RACHECK.

PROFILE=*profile addr*

ENTITY=(*resource name addr*,NONE)

profile addr: Rx-type address or register (2) - (12).

Note: PROFILE or ENTITY is required on either the list or the execute form of the macro.

,VOLSER=*vol addr*

Note: VOLSER is required on either the list or the execute form of the macro, but only for CLASS='DATASET' and DSTYPE not equal to M when a discrete profile name is used. If required, VOLSER must be specified on either the list or the execute form of the macro.

,CLASS=*class name addr*

class name addr: Rx-type address or register (2) - (12).

,RELEASE=*number*

,RELEASE=(,CHECK)

,RELEASE=*(number,*CHECK)

number: 1.8.2, 1.8.1, 1.8, 1.7, or 1.6

Default: RELEASE=1.6

,ATTR=READ

,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER ,ATTR=*reg*	*reg*: Register (2) - (12)
,DSTYPE=N ,DSTYPE=V ,DSTYPE=M ,DSTYPE=T	
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12).
,LOG=ASIS ,LOG=NOFAIL ,LOG=NONE ,LOG=NOSTAT	
,OLDVOL=*old vol addr*	*old vol addr*: Rx-type address or register (2) - (12).
	applname addr: Rx-type address or register (2) - (12).
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12).
,ACCLVL=(*access value addr*)	*access value addr*: Rx-type address or register (2) - (12).
,ACCLVL=(*access value addr*)	Rx-type address or register (2) - (12)
,RACFIND=YES ,RACFIND=NO	
,GENERIC=YES ,GENERIC=ASIS	
,FILESEQ=*reg*	*reg*: Register (2) - (12)
,FILESEQ=*number*	*number*: 1-9999
,TAPELBL=STD ,TAPELBL=BLP ,TAPELBL=NL	
,STATUS=NONE	

,STATUS=ERASE

,USERID=*userid addr* — *userid addr*: Rx-type address or register (2) - (12).

,GROUPID=*groupid addr* — *groupid addr*: Rx-type address or register (2) - (12).

,MF=(E,*ctrl addr*) — *ctrl addr*: Rx-type address or register (1) or (2) - (12).

The parameters are explained under the standard form of the RACHECK macro with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACHECK macro instruction.

,RELEASE=(*number*,CHECK)
,RELEASE=1.6|1.7|1.8|1.8.1|1.8.2
,RELEASE=(,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 8 on page 242.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACHECK macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACINIT: Identify a RACF-Defined User

The RACINIT macro is used to provide RACF user identification and verification. The macro instruction identifies a user and verifies that the user is defined to RACF and has supplied a valid password.

On z/VM, you can use the RACINIT macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACINIT macro from a user's machine.

RACINIT (Standard Form)

The standard form of the RACINIT macro is written as follows:

name — *name*: Symbol. Begin *name* in column 1.

␣ — One or more blanks must precede RACINIT.

RACINIT

␣	One or more blanks must follow RACINIT.
USERID=*user ID addr*	*user ID addr*: A-type address or register (2) - (12).
,PASSWRD=*password addr*	*password addr*: A-type address or register (2) - (12).
,NEWPASS=*new password addr*	*new password addr*: A-type address or register (2) - (12).
,GROUP=*group addr*	**Default:** zero.
,PGMNAME=*programmer name addr*	*programmer name addr*: A-type address or register (2) - (12).
,ACTINFO=*account addr*	*account addr*: A-type address or register (2) - (12).
,TERMID=*terminal addr*	*terminal addr*: A-type address or register (2) - (12).
,JOBNAME=*jobname addr*	*jobname addr*: A-type address or register (2) - (12).
,ENVIR=CREATE	**Default:** ENVIR=CREATE
,ENVIR=CHANGE ,ENVIR=DELETE	**Note:** 1. ENVIR=CHANGE may not be specified with USERID=, PASSWRD=, NEWPASS=, ACTINFO=, PGMNAME=, or TERMID= parameters. 2. ENVIR=DELETE may not be specified with APPL=, USERID=, PASSWRD=, NEWPASS=, GROUP=, ACTINFO=, PGMNAME=, or TERMID= parameters.
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12).
,APPL=*'applname'*	*applname*: 1- to 8-character name
,APPL=*applname addr*	*applname addr*: A-type address or register (2) - (12).
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12).

,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255.
,PASSCHK=YES ,PASSCHK=NO	**Default:** PASSCHK=YES
,ENCRYPT=YES ,ENCRYPT=NO	**Default:** ENCRYPT=YES
,RELEASE=*number*	**Default:** RELEASE=1.6
,STAT=ASIS ,STAT=NO	**Default:** STAT=ASIS
,LOG=ASIS ,LOG=ALL	**Default:** LOG=ASIS

The parameters are explained as follows:

USERID=*user ID addr*
specifies the user identification of the user who has entered the system. The address points to a 1-byte length field, followed by the user ID, which can be up to 8 characters in length.

Application considerations: When verifying a user ID, be sure to validate that it contains only characters that are alphabetic, numeric, # (X'7B'), @ (X'7C'), or $ (X'5B') and is 1-8 characters in length. Additionally, you must change the user ID to uppercase.

,PASSWRD=*password addr*
specifies the currently defined password of the user who has entered the system. The address points to a 1-byte length field, followed by the password, which can be up to eight characters in length.

Application considerations: When verifying a new password, validate that it is 1-8 characters in length. If SETROPTS MIXEDCASE is not in effect, you must change the password to uppercase. Avoid performing any other checking of character content, letting the security product determine the validity of the password.

,NEWPASS=*new password addr*
specifies the password that is to replace the user's currently defined password. The address points to a 1-byte length field, followed by the password, which can be up to eight characters in length.

Application considerations: When verifying a new password, validate that it is 1-8 characters in length. If SETROPTS MIXEDCASE is not in effect, you must change the password to uppercase. Avoid performing any other checking of character content, letting the security product determine the validity of the password.

,GROUP=*group addr*
specifies the group specified by the user who has entered the system. The address points to a 1-byte length field, followed by the group name, which can be up to eight characters in length.

,PGMNAME=*programmer name addr*
specifies the address of the name of the user who has entered the system. This 20-byte area is passed to the RACINIT installation exit routine; it is not used by the RACINIT routine.

,ACTINFO=*account addr*
specifies the address of a field containing accounting information. This 144-byte area is passed to the RACINIT installation exit routine; it is not used by the RACINIT routine. The accounting field, if supplied, should have the following format:

- The first byte of field contains the number (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.

,TERMID=*terminal addr*
specifies the address of the identifier for the terminal through which the user is accessing the system. The address points to an 8-byte area containing the terminal identifier. Information specified by TERMID= on an ENVIR=CREATE may be attached to the created ACEE and used in subsequent RACF processing. RACF does not make its own copy of this area when attaching this information to the created ACEE. This area must not be explicitly freed prior to the deletion of the ACEE. For the same reason, the area must reside in a non-task-related storage subpool so that implicit freeing of the area does not occur.

,JOBNAME=*jobname addr*
specifies the address of the job name of a background job. The address points to an 8-byte area containing the job name (left-justified and padded with blanks if necessary). The JOBNAME parameter is used by RACINIT during authorization checking to verify the user's authority to submit the job. It is passed to the installation exit routine.

On z/VM, a background job may be used to submit a job to a batch service machine for processing.

,ENVIR=CREATE
,ENVIR=CHANGE
,ENVIR=DELETE
specifies the action to be performed by the user-initialization component regarding the ACEE:

- CREATE: The user should be verified and an ACEE created.
- CHANGE: The ACEE should be modified according to other parameters specified on RACINIT. You can change only the connect group with this option.
- DELETE: The ACEE should be deleted. This parameter should be used only if a previous RACINIT has completed successfully.

Attention: IBM recommends issuing a RACINIT,ENVIR=DELETE to delete only an ACEE that you created. See “Special Considerations for Changing or Deleting an ACEE” on page 255 for alternative options.

,INSTLN=*parm list addr*
specifies the address of an area containing parameter information meaningful to the RACINIT installation exit routine. This area is passed to the installation exit when the exit routine is given control from the RACINIT routine.

The INSTLN parameter can be used by an installation having a user verification or job initiation application, and wanting to pass information from one installation module to the RACINIT installation exit routine.

,APPL=‘*applname*’
,APPL=*applname addr*
specifies the name of the application issuing the RACINIT. If an address is specified, the address must point to an 8-byte application name, left-justified and padded with blanks if necessary.

,ACEE=*acee addr*
specifies the address of the ACEE.

For ENVIR=DELETE: specifies the address of a fullword that contains the address of the ACEE to be deleted. If ACEE= is not specified, and the TCBSENV field for the task using the RACINIT is nonzero, the ACEE pointed to by the TCBSENV is deleted, and TCBSENV is set to zero. If the TCBSENV and ASXBSENV fields both point to the same ACEE, 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.

For ENVIR=CHANGE: specifies the address of a fullword that contains the address of the ACEE to be changed. If ACEE= is not specified, and the TCBSENV field for the task using the RACINIT is nonzero, the ACEE pointed to by the TCBSENV is changed. If TCBSENV is 0, the ACEE pointed to by ASXBSENV is changed.

For ENVIR=CREATE: specifies the address of a fullword into which the RACINIT function places the address of the ACEE created. If an ACEE is not specified, the address of the newly created ACEE is stored in the TCBSENV field of the task control block. If the ASXBSENV field is set to binary zeros, the new ACEE address is also stored in the ASXBSENV field of the ASXB. If the ASXBSENV field is nonzero, it is not modified. The TCBSENV field is set unconditionally.

Note: If you omit USERID, GROUP, and PASSWRD and if you code ENVIR=CREATE or if ENVIR=CREATE is used as the default, you receive a return code of X'00' and obtain an ACEE that contains an * (X'5C') in place of the user ID and group name.

,SUBPOOL=*subpool number*
specifies the storage subpool from which the ACEE and related storage are obtained. The value of subpool can be literally specified or passed through a register. When literally specified, the valid values are 0 through 255. When you use a register, the subpool number is the value of the least significant byte in the register.

,PASSCHK=YES
,PASSCHK=NO
specifies whether the user's password is to be verified. PASSCHK=YES specifies that RACINIT verifies the user's password. PASSCHK=NO specifies that the user's password is not verified.

,ENCRYPT=YES
,ENCRYPT=NO
specifies whether or not RACINIT encodes the old password and the new password.

YES signifies that the data specified by the PASSWRD and NEWPASS keywords are not preencoded. RACINIT encodes the data before storing it in the user profile or using it to compare against stored data. ENCRYPT=YES is the default for this keyword.

NO signifies that the data specified by the PASSWRD and NEWPASS keywords are already encoded. RACINIT bypasses the encoding of this data before storing it in or comparing it against the user profile.

Note: If a RACF password is encrypted using KDFAES, then the data that is specified by the PASSWRD= keyword must be encoded using the DES method to be evaluated successfully. If SETROPTS PASSWORD(ALGORITHM(KDFAES)) is active, then the data that is specified by the NEWPASS= keyword must be encoded using the DES method to create a new password that is correctly evaluated.

,RELEASE=1.6|1.7|1.8|1.8.1
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 9 on page 254.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time.

,STAT=ASIS|NO
specifies whether the statistics controlled by the installation's options on the RACF SETROPTS command are to be maintained or ignored for this execution of RACINIT. This parameter also controls whether a message is to be issued when the logon is successful.

Note: Messages are always issued if the RACINIT processing is unsuccessful.

If STAT=ASIS is specified or taken by default, the messages and statistics are controlled by the installation's current options on the RACF SETROPTS command.

If STAT=NO is specified, the statistics are not updated. And if the logon is successful, no message is issued.

The default is STAT=ASIS.

,LOG=ASIS|ALL
specifies when log records are to be generated.

If LOG=ASIS is specified or defaulted to, only those attempts to create an ACEE that fails generate RACF log records.

If LOG=ALL is specified, any request to create an ACEE, regardless of whether it succeeds or fails, generates a RACF log record. The default is LOG=ASIS.

Parameters for RELEASE=1.6 through 1.8.1

The RELEASE values for which a specific parameter is valid are marked with an X.

Table 9. RACINIT Parameters for RELEASE=1.6 through 1.8.1

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
ACEE=	X	X	X
ACCTINFO=	X	X	X
APPL=	X	X	X
ENCRYPT=	X	X	X
ENVIR=	X	X	X
GROUP=	X	X	X
INSTLN=	X	X	X
JOBNAME=	X	X	X
LOG=		X	X
NEWPASS=	X	X	X
PASSCHK=	X	X	X
PASSWRD=	X	X	X
PGMNAME=	X	X	X
RELEASE=	X	X	X
STAT=		X	X
SUBPOOL=	X	X	X
TERMID=	X	X	X

Table 9. RACINIT Parameters for RELEASE=1.6 through 1.8.1 (continued)

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
USERID=	X	X	X

Special Considerations for Changing or Deleting an ACEE

IBM recommends that you delete only an ACEE that you created. Issuing a RACINIT with ENVIR=DELETE specified to delete the existing ACEE can lead to problems if you are not the one who created that environment. The issuer of the ENVIR=CREATE that built the ACEE may have saved a pointer to it and may be expecting it to be available later in processing. Note that this is the case for the initiator's ACEE. Also, if you delete an ACEE, you may lose tables anchored off that ACEE that are needed later in RACF processing. See *z/VM: RACF Security Server Diagnosis Guide* for overview diagrams of ACEEs and related control blocks that can be useful when diagnosing problems.

Note: When you delete an ACEE that has a third-party ACEE attached, the RACINIT pre- or post-exits get control again for the third-party ACEE as well as for the original ACEE being deleted.

If you make a copy of the ACEE and update fields, avoid passing it to RACHECK or RACDEF. These services anchor tables off the ACEE and refresh these tables when required. If you update fields in a copy, the original ACEE then contains invalid pointers that result in abends when the original is used or deleted.

If you need to delete or change an ACEE that you did not create, you can use one of the following methods.

- **Change the values in the current ACEE:**

  Issue RACINIT with ENVIR=CHANGE to change the values in the current ACEE.

- **Create, anchor, and delete a third-party ACEE:**

  Issue RACHECK with USERID= and GROUPID= causing RACF to create, anchor, and delete a third-party ACEE internally.

Return Codes and Reason Codes

If the return codes and reason codes you are receiving are not discussed in this macro, refer to the return codes and reason codes described with RACROUTE REQUEST=VERIFY.

When control is returned, register 15 contains one of the following return codes, and register 0 may contain a reason code.

Hexadecimal Code
Meaning

00
RACINIT has completed successfully.

04
The user profile is not defined to RACF.

08
The password is not authorized.

0C
The password has expired.

10
The new password is invalid.

14
The user is not defined to the group.

18
RACINIT was failed by the installation exit routine.

1C
The user's access has been revoked.

20
RACF is not active.

24
The user's access to the specified group has been revoked.

30
The user is not authorized to use the terminal. Register 0 contains one of the following reason codes:

00
Indicates a normal completion.

04
Indicates the user is not authorized to access the system on this day, or at this time of day.

08
Indicates the terminal may not be used on this day, or at this time of day.

34
The user is not authorized to use the application.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACINIT macro; however, the list form of the macro does not have the proper RELEASE parameter. Macro processing terminates.

Example 1

Operation: Use the standard form of the macro to do the following:

- Create an ACEE for the user ID and its default group.
- Verify that the user named USERNAME is a valid user.
- Verify that the password called PASSWORD is valid.

```
RACINIT ENVIR=CREATE,USERID=USERNAME,PASSWRD=PASSWORD
```

Example 2

Operation: Use the standard form to do the following:

- Verify that the user named USERNAME is a valid user.
- Verify that the group named GROUPNAM is a valid group.
- Verify that USERNAME is defined to the group.
- Create an ACEE for the user and group and put its address in ACEEANCH.
- Specify that the user's password is not required.

```
RACINIT ENVIR=CREATE,USERID=USERNAME,GROUP=GROUPNAM,ACEE=ACEEANCH,    X
          PASSCHK=NO
```

Example 3

Operation: Use the standard form of the macro to delete the ACEE of the current task or address space, or both.

```
RACINIT ENVIR=DELETE
```

RACINIT (List Form)

The list form of the RACINIT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACINIT	One or more blanks must precede RACINIT.
␣	One or more blanks must follow RACINIT.
USERID=*user ID addr*	*user ID addr*: A-type address.
,PASSWRD=*password addr*	*password addr*: A-type address.
,NEWPASS=*new password addr*	*new password addr*: A-type address.
,GROUP=*group addr*	*group addr*: A-type address.
,PGMNAME=*programmer name addr*	*programmer name addr*: A-type address.
,ACTINFO=*account addr*	*account addr*: A-type address.
,TERMID=*terminal addr*	*terminal addr*: A-type address.
,JOBNAME=*jobname addr*	*jobname addr*: A-type address.
,ENVIR=CREATE	**Default:** ENVIR=CREATE

,ENVIR=CHANGE ,ENVIR=DELETE	**Note:** 1. ENVIR=CHANGE may not be specified with USERID=, PASSWRD=, NEWPASS=, ACTINFO=, PGMNAME=, or TERMID= parameters. 2. ENVIR=DELETE may not be specified with APPL=, USERID=, PASSWRD=, NEWPASS=, GROUP=, ACTINFO=, PGMNAME=, or TERMID= parameters.
,INSTLN=*parm list addr*	*parm list addr*: A-type address.
,APPL=*'applname'* ,APPL=*applname addr*	*applname*: 1- to 8-character name *applname addr*: A-type address.
,ACEE=*acee addr*	*acee addr*: A-type address.
,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255.
,PASSCHK=YES ,PASSCHK=NO	**Default:** PASSCHK=YES
,ENCRYPT=YES ,ENCRYPT=NO	**Default:** ENCRYPT=YES
,RELEASE=*number*	**Default:** RELEASE=1.6
,STAT=ASIS ,STAT=NO	**Default:** STAT=ASIS
,LOG=ASIS ,LOG=ALL	**Default:** LOG=ASIS
,MF=L	

The parameters are explained under the standard form of the RACINIT macro instruction with the following exception:

,MF=L
specifies the list form of the RACINIT macro instruction.

RACINIT (Execute Form)

The execute form of the RACINIT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACINIT	One or more blanks must precede RACINIT.
␣	One or more blanks must follow RACINIT.
USERID=*user ID addr*	*user ID addr*: Rx-type address or register (2) - (12).
,PASSWRD=*password addr*	*password addr*: Rx-type address or register (2) - (12).
,NEWPASS=*new password addr*	*new password addr*: Rx-type address or register (2) - (12).
,GROUP=*group addr*	*group addr*: Rx-type address or register (2) - (12).
,PGMNAME=*programmer name addr*	*programmer name addr*: Rx-type address or register (2) - (12).
,ACTINFO=*account addr*	*account addr*: Rx-type address or register (2) - (12).
,TERMID=*terminal addr*	*terminal addr*: Rx-type address or register (2) - (12).
,JOBNAME=*jobname addr*	*jobname addr*: Rx-type address or register (2) - (12).
,ENVIR=CREATE ,ENVIR=CHANGE ,ENVIR=DELETE	**Note:** 1. ENVIR=CHANGE may not be specified with USERID=, PASSWRD=, NEWPASS=, ACTINFO=, PGMNAME=, or TERMID= parameters. 2. ENVIR=DELETE may not be specified with APPL=, USERID=, PASSWRD=, NEWPASS=, GROUP=, ACTINFO=, PGMNAME=, or TERMID= parameters.
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12).

,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12).
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12).
,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255.
,PASSCHK=YES ,PASSCHK=NO	
,ENCRYPT=YES ,ENCRYPT=NO	
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=*(number,*CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6
,STAT=ASIS ,STAT=NO	
,LOG=ASIS ,LOG=ALL	
,MF=(E,*ctrl addr)*	*ctrl addr*: Rx-type address or register (1) or (2) - (12).

The parameters are explained under the standard form of the RACINIT macro instruction with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACINIT macro, using a remote, control-program parameter list.

,RELEASE=1.6|1.7|1.8|1.8.1
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time. If you specify a parameter with an incompatible release level, the parameter will not be accepted by macro processing. An error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 9 on page 254.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACINIT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

RACLIST: Build In-Storage Profiles

RACLIST is used to build in-storage profiles for RACF-defined resources. RACLIST processes only those resources described by class descriptors. The primary advantage of using the RACLIST macro is to use the resource-grouping function and to improve resource-authorization-checking performance.

You can use the RACLIST macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACLIST macro from a user's machine.

Note:

1. Only callers in 24-bit addressing mode can issue this macro. Callers executing in 31-bit addressing mode who want to use the RACLIST function can code the RACROUTE macro.
2. You can use the RACLIST macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACLIST macro from a user's machine.

RACLIST (Standard Form)

The standard form of the RACLIST macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACLIST	One or more blanks must precede RACLIST.
␣	One or more blanks must follow RACLIST.
CLASS=*'class name'* CLASS=*class name addr*	*class name*: 1-to 8-character class name *class name addr*: A-type address or register (2) - (12)
,LIST=*list addr*	*list addr*: A-type address or register (2) - (12)
,ACEE=*acee addr*	*acee addr*: A-type address or register (2) - (12).
,INSTLN=*parm list addr*	*parm list addr*: A-type address or register (2) - (12).
,APPL=*'applname'* ,APPL=*applname addr*	*applname*: 1- to 8-character name *applname addr*: A-type address or register (2) - (12).

,ENVIR=CREATE ,ENVIR=DELETE	**Default:** ENVIR=CREATE
,OWNER=YES ,OWNER=NO	**Default:** OWNER=NO
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6

The parameters are explained as follows:

CLASS='*class name*'
CLASS=*class name addr*
specifies that RACLIST is to build an in-storage profile for the resources of the specified class. If an address is specified, the address must point to an 8-byte field containing the class name, left-justified and padded with blanks if necessary. The class name must be defined by a class descriptor; if not, the class is not considered to be defined.

,LIST=*addr*
specifies the address of a list of resource names for which RACLIST is to build the in-storage profiles. The list consists of a 2-byte field containing the number of the names in the list, followed by one or more variable-length names. Each name consists of a 1-byte length field, which is the length of the name, followed by the name. A zero in the 2-byte field causes the operand to be omitted. If LIST= is omitted, in-storage profiles are built for all the profiles defined to RACF in the given class as well as each member for a resource grouping associated with the specified class.

Note: This operand can be specified only with ENVIR=CREATE. If ENVIR=DELETE is specified, the RACLIST macro issues a return code of 18.

,ACEE=*acee addr*
specifies the address of the ACEE. The ACEE points to the in-storage profiles. If an ACEE is not specified, RACF uses the TASK ACEE pointer in the extended TCB called the TCBSENV. Otherwise, or if the TASK ACEE pointer is zero, RACF uses the main ACEE to obtain the list of the in-storage profiles. The main ACEE is pointed to by the ASXBSENV field of the address-space extension block. If an ACEE is not specified and there is no main ACEE, the in-storage profiles are not constructed.

,INSTLN=*parm list addr*
specifies the address of an area that contains parameter information for the RACLIST installation exit. The address is passed to the installation exit when the RACLIST routine gives control to the exit. An application or an installation program can use the INSTLN parameter to pass information to the RACLIST installation exit.

,APPL='*applname*'
,APPL=*applname addr*
specifies the name of the application requesting the authorization checking. This information is not used for the authorization-checking process but is made available to the installation exit or exits. If an address is specified, it should point to an 8-byte area containing the application name, left justified and padded with blanks if necessary.

,ENVIR=CREATE
,ENVIR=DELETE
specifies the action to be performed by the RACLIST macro.

CREATE: In-storage profiles for the specified class are to be built. The RACLIST function issues a return code of 18, if an in-storage list currently exists for the specified class.

DELETE: The in-storage profiles for the specified class are to be freed. If class is not specified, the in-storage profiles for all classes are freed.

Note: It is the responsibility of the user issuing the RACLIST macro to assure that no multitasking that results in the issuing of a RACHECK, FRACHECK, RACINIT, or RACLIST macro occurs at the same time that the RACLIST occurs.

,OWNER=YES
,OWNER=NO

specifies that the resource owner is to be placed in the profile access list with the ALTER authority. If the OWNER= operand is omitted, the default is NO.

,RELEASE=1.6|1.7|1.8|1.8.1

specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error message is issued at assembly time.

For the parameters that are valid for RELEASE=1.6 and later, see Table 10 on page 263.

Parameters for RELEASE=1.6 through 1.8.1

The RELEASE values for which a specific parameter is valid are marked with an X.

Table 10. RACLIST Parameters for RELEASE=1.6 through 1.8.1

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
ACEE=	X	X	X
APPL=	X	X	X
CLASS=	X	X	X
ENVIR=	X	X	X
INSTLN=	X	X	X
LIST=	X	X	X
OWNER=	X	X	X
RELEASE=	X	X	X

Return Codes and Reason Codes

If the return codes and reason codes you are receiving are not discussed in this macro, refer to “Return Codes and Reason Codes” on page 131.

When control is returned, register 15 contains one of the following return codes, and register 0 may contain a reason code.

Hexadecimal Code
Meaning

00

RACLIST function completed successfully.

0

Delete request successful. Create request successful and profiles were listed.

4
Create request successful but no profiles were listed.

04
Unable to perform the requested function. Register 0 contains additional codes as follows:

0
Unable to establish an ESTAE environment.

1
The function code (the third byte of the parameter list) does not represent a valid function. 01 represents the RACF manager; 02 represents the RACLIST macro.

08
The specified class is not defined to RACF.

0C
An error was encountered during RACLIST processing.

10
RACF or the resource class is not active, or both.

14
RACLIST installation exit error occurred.

18
Parameter-list error.

1C
RACF is not installed, or an insufficient level of RACF is installed.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACLIST macro; however, the list form of the macro does not have the proper RELEASE parameter. Macro processing terminates.

Note: If the resource class specified by the CLASS= operand is inactive, RACLIST does not build the in-storage profiles and a code of 0C is returned. If the resource-group class is not active, RACLIST builds an in-storage profile but only from the individual resource profiles; resource-group profiles are ignored.

Example 1

Operation: Use the standard form of the macro to build in-storage profiles for all the profiles in the DASDVOL class and chain them off the ACEE whose address is pointed to by ACEEADDR.

```
RACLIST CLASS='DASDVOL',ACEE=ACEEADDR,ENVIR=CREATE
```

Example 2

Operation: Use the standard form of the macro to build in-storage profiles for all the profiles whose names are in a list named PROFLIST and DASDVOL class. Chain them from the task ACEE or address space ACEE.

```
RACLIST CLASS='DASDVOL',LIST=PROFLIST,ENVIR=CREATE
   .
   .
   .
PROFLIST  DS   0CL35
PROFNUM   DC   XL2'0005'
PROF1     DC   AL1(6),CL6'DASD01'
PROF2     DC   AL1(6),CL6'DASD02'
PROF3     DC   AL1(5),CL5'DASDA'
PROF4     DC   AL1(5),CL5'DASDB'
PROF5     DC   AL1(6),CL6'MYDASD'
```

Example 3

Operation: Use the standard form of the macro to delete the in-storage profiles for the DASDVOL class.

```
RACLIST CLASS=DASDVOL,ENVIR=DELETE
```

RACLIST (List Form)

The list form of the RACLIST macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACLIST	One or more blanks must precede RACLIST.
␣	One or more blanks must follow RACLIST.
CLASS=*'class name'* CLASS=*class name addr*	*class name*: 1- to 8-character class name *class name addr*: A-type address
,LIST=*list addr*	*list addr*: A-type address
,ACEE=*acee addr*	*acee addr*: A-type address.
,INSTLN=*parm list addr*	*parm list addr*: A-type address.
,APPL=*'applname'* ,APPL=*applname addr*	*applname*: 1- to 8-character name *applname addr*: A-type address
,ENVIR=CREATE ,ENVIR=DELETE	**Default:** ENVIR=CREATE
,OWNER=YES ,OWNER=NO	**Default:** OWNER=NO
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6

,MF=L

The parameters are explained under the standard form of the RACLIST macro with the following exception:

,MF=L
specifies the list form of the RACLIST macro instruction.

RACLIST (Execute Form)

The execute form of the RACLIST macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACLIST	One or more blanks must precede RACLIST.
␣	One or more blanks must follow RACLIST.
CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12).
,LIST=*list addr*	*list addr*: Rx-type address or register (2) - (12).
,ACEE=*acee addr*	*acee addr*: Rx-type address or register (2) - (12).
,INSTLN=*parm list addr*	*parm list addr*: Rx-type address or register (2) - (12).
,APPL=*applname addr*	*applname addr*: Rx-type address or register (2) - (12).
,ENVIR=CREATE ,ENVIR=DELETE	
,OWNER=YES ,OWNER=NO	
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=*(number,*CHECK)	*number*: See Standard Form **Default:** RELEASE=1.6

,MF=(E,*ctrl addr*) *ctrl addr*: Rx-type address or register (2) - (12).

The parameters are explained under the standard form of the RACLIST macro with the following exceptions:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACLIST macro instruction, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by macro processing, and an error is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 10 on page 263.

The default is RELEASE=1.6. When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACLIST macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

Limited Function RACROUTE on z/VM (RACROUTE REQUEST=AUTH with RELEASE=1.8.2 specified)

Note:
The limited-function RACROUTE macro for z/VM described in this section is only invoked when RELEASE=1.8.2 is specified on a RACROUTE invocation. To use the full-function RACROUTE macro on z/VM, you must specify RELEASE=1.9 or later. To use the full-function RACROUTE, see “RACROUTE: Router Interface” on page 8. **Even with the RACF 1.9 (or later) product installed, RELEASE=1.8.2 can still be specified to invoke limited-function RACROUTE.**

With Release 1.8.2 on z/VM, you can use the RACROUTE macro instruction, REQUEST=AUTH, which allows a virtual machine to invoke authorization checking on RACF-defined general resources. The other request types supported on z/OS RACROUTE are not supported on z/VM RACROUTE when RELEASE=1.8.2 is specified.

RACROUTE on z/VM supports both standard authorization checking and third-party authorization checking. Third-party RACROUTE REQUEST=AUTH allows you to do authorization checking on behalf of another user. See “Example 1” on page 270.

To perform a standard RACROUTE REQUEST=AUTH, an invoker needs only a CP privilege class of G; however, to invoke the RACROUTE to perform a third-party RACROUTE REQUEST=AUTH, an invoker must have a CP Privilege Class of A, B, C, D, E, or F. For instance, a resource manager such as a tape-management system could employ the third-party RACROUTE REQUEST=AUTH to determine whether a specific user is authorized to the RACF-protected resources that the tape-management system controls. However, for that manager to issue a third-party RACROUTE REQUEST=AUTH, it must run in a privileged machine.

Note: In addition to the CP privilege class requirements described, the caller of limited-function RACROUTE must have a least READ authority to the ICHCONN profile in the FACILITY class.

To indicate whether the invoker or the user ID passed by the invoker has the authority to access a resource, the RACF service machine issues a a RACROUTE REQUEST=AUTH and sends the return code to the invoker.

RACROUTE only processes requests in the general-resource classes; for example, VMMDISK, VMCMD, FACILITY, and TAPEVOL. The maximum entity name length is 39. RACROUTE does not process requests in the DATASET class.

Note: Various RACF functions invoked by RACROUTE require that you specify the CLASS parameter, and that the specified CLASS be active. With few exceptions, for the IBM-supplied portion of the table, the class specified on the CLASS parameter **must** be active for the RACROUTE macro to invoke RACF. In the case of the installation-supplied portion of the table, there are no exceptions; the class specified on the CLASS parameter **must** be active for the RACROUTE macro to invoke RACF.

RACROUTE on z/VM (Standard Form)

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=AUTH	REQUEST is a required keyword for z/VM expansion.
,WORKA=*work area addr*	*work area addr*: A-type address or register (2) - (12).
RELATED=*value*	*value*: Any valid macro keyword specified
,USERID='*userid*' ,USERID=*userid addr*	*userid*: 1- to 8-character user ID *userid addr*: A-type address or register (2) - (12)
,MF=S	Generates the standard form of the macro
,ENTITY=*(resource name addr)*	*resource name addr*: A-type address or register (2) - (12)
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER	**Default:** READ
,ATTR=*register*	Register (2) - (12)
,CLASS='*class name*'	*class name*: 1- to 8-character class name

,CLASS=*class name addr*	*class name address*: A-type address or register (2) - (12)
,RELEASE=*number*	1.8.2 allows z/VM expansion.

The parameters are explained as follows:

REQUEST=AUTH
specifies that an authorization request is to be performed.

,WORKA=*work area addr*
specifies the address of a 64-byte work area used by the macro to build the diagnosis parameter list. The storage where the RACROUTE macro builds the parameter list must be:

- Not Read-protected
- Doubleword-aligned
- Contained within a page.

,RELATED=*value*
specifies information used to self-document macro instructions by relating functions or services to corresponding functions or services. The format and contents of the information specified is at the discretion of the user, and can be any valid coding value.

,USERID='*userid*'
,USERID=*userid addr*
specifies the user ID that RACF uses to perform third-party authorization checks. If USERID is specified when the caller invokes RACROUTE on z/VM, RACF performs authorization checking on the ACEE of the USERID. If USERID is not specified, RACF performs authorization checking on the ACEE of the issuer of RACROUTE.

,MF=S
specifies the standard form of the macro.

,ENTITY=(*resource name addr*)
specifies that RACF authorization checking is to be performed for the resource whose name is pointed to by the specified address. The resource name is a 6-byte volume serial number for CLASS='DASDVOL' or CLASS='TAPEVOL'. The length of all other resource names is determined from the class-descriptor tables. The name must be left-justified and padded with blanks. The actual resource-name length is determined by the class-descriptor table.

,ATTR=
,ATTR=*reg*
specifies the access authority of the user or group permitted access to the resource for which RACF authorization checking is to be performed:

READ
RACF user or group can open the resource only to read.

UPDATE
RACF user or group can open the resource to write or read.

CONTROL
For VSAM data sets, RACF user or group has authority equivalent to the VSAM control password. For non-VSAM data sets and other resources, RACF user or group has UPDATE authority.

ALTER
RACF user or group has total control over the resource.

If a register is specified, the register must contain one of the following codes in the low-order byte of the register:

X'02' READ
X'04' UPDATE

X'08' CONTROL
X'80' ALTER.

,CLASS='*class name*'
,CLASS=*class name addr*
specifies that RACF authorization checking is to be performed for a resource of the specified class. If an address is specified, the address must point to a 1-byte field indicating the length of the class name, followed by the class name.

,RELEASE=*1.8.2*
specifies the release number. To use the limited-function RACROUTE in the z/VM environment, you must specify Release=1.8.2. To use the full-function RACROUTE macro on z/VM, specify RELEASE=1.9 or later. To use the full-function RACROUTE, see "RACROUTE: Router Interface" on page 8.

Return Codes and Reason Codes

When you receive control back after the execution of the RACROUTE macro, register 15 contains the return code (such as 0, 4, 8) from RACROUTE. In addition, the first two words of the RACROUTE parameter list contain the RACF return code and reason code, respectively. You can map the parameter list using the ICHSAFP macro, based on the address of your MF=L form of RACROUTE.

When control is returned, register 15 contains one of the following return codes. These same codes will also be in the return code field (SAFPRRET) in the RACROUTE parameter list.

Hexadecimal Code
Meaning

00
The authorization request is allowed.

04
The authorization request is deferred.

08
The authorization request has failed or RACF is not available.

0C
The parameter list length is incorrect.

282
The ATTR or class name is incorrect.

Note: The RACROUTE support in the z/VM environment does not return the RACF reason code. Therefore, both the reason code field in the RACROUTE parameter list and register 0 are set to zero when control is returned.

Example 1

Operation: A CP-privileged resource manager invokes the RACROUTE (REQUEST=AUTH) as a third-party RACHECK to determine whether z/VM user SUE is RACF-authorized to access another user's (TOM's) 191 minidisk.

```
RACROUTE  REQUEST=AUTH,CLASS=CLASSNL,ENTITY=ENTITYNA,      X
          RELEASE=1.8.2,MF=S,WORKA=WORK,ATTR=READ,         X
          USERID=VMUSER
  .
  .
  .
ENTITYNA  DC CL39'TOM.191'     * Entity
CLASSNL   DC XL1'07'           * Class name length
CLASSN    DC CL8'VMMDISK'      * Class name
VMUSER    DC CL8'SUE'          * Requesting z/VM user ID
          DS OD                * Ensure double word alignment
WORK      DS CL64              * Storage for macro expansion
```

RACROUTE on z/VM (List Form)

On the LIST form of the macro invocation, do not specify registration of the keywords. The LIST form is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=AUTH	REQUEST is a required keyword for z/VM expansion.
,WORKA=*work area addr*	*work area addr*: A-type address
,RELATED=*value*	*value*: Any valid macro keyword specified
,USERID=*userid addr*	*userid addr*: A-type address
,MF=L	Generates the list form of the macro
,ENTITY=(*resource name address*)	*resource name addr*: A-type address
,ATTR=READ	**Default:** READ
,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER	
,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address
,RELEASE=*number*	1.8.2 allows z/VM expansion.

The parameters are explained under the standard form of the RACROUTE macro with the following exception:

,MF=L
specifies the list form of the RACROUTE macro instruction.

RACROUTE on z/VM (Execute Form)

The execute form of the RACROUTE macro updates the list form of the macro, builds the diagnose RACROUTE parameter list in the work area, and issues the diagnose to invoke RACF services. It is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACROUTE	One or more blanks must precede RACROUTE.
␣	One or more blanks must follow RACROUTE.
REQUEST=AUTH	REQUEST is a required keyword for z/VM expansion.
,WORKA=*work area addr*	*work area addr*: Rx-type address or register (2) - (12).
,RELATED=*value*	*value*: Any valid macro keyword specified
,USERID=*userid addr*	*userid addr*: Rx-type address or register (2) - (12)
,MF=(E,*ctrl addr*)	*addr*: Rx-type address or register (2) - (12)
,ENTITY=(*resource name addr*)	*resource name addr*: Rx-type address or register (2) - (12)
,ATTR=READ ,ATTR=UPDATE ,ATTR=CONTROL ,ATTR=ALTER	
,ATTR=*reg*	*reg*: Register (2) - (12)
,CLASS=*class name addr*	*class name address*: Rx-type address or register (2) - (12)
,RELEASE=number	1.8.2 allows z/VM expansion.

The parameters are explained under the standard form of the RACROUTE macro with the following exception:

,MF=(E,*ctrl addr*)
specifies the execute form of the RACROUTE macro instruction.

RACROUTE: SAF Router Interface

The RACROUTE macro is used to invoke the system-authorization-facility (SAF) router. Depending on how your installation has written the SAF router exit and on whether RACF is present, the SAF router directs control to the RACF router. If RACF is active, the RACF router invokes RACF.

Note: Various RACF functions invoked by RACROUTE require that you specify the CLASS parameter, and that the specified CLASS be active. With few exceptions, for the IBM-supplied portion of the table, the class specified on the CLASS parameter **must** be active for the RACF router to invoke RACF. In the case of the installation-supplied portion of the table, there are no exceptions; the class specified on the CLASS parameter **must** be active for the RACF router to invoke RACF.

You can use RACROUTE to access the functions that are provided by the following RACF macros: RACDEF, RACINIT, RACXTRT, RACLIST, RACHECK, and FRACHECK. In coding the RACROUTE macro to access a particular RACF macro function, you must also use the necessary parameters from that macro on the RACROUTE macro instruction. For example, if you code RACROUTE to access the RACHECK function, you must code REQUEST=AUTH as well as any other required and optional parameters you need from the RACHECK macro. RACROUTE validates that only the parameters applicable to the RACHECK macro have been coded.

Note:

1. For RACF Version 1 Release 6 and earlier, all parameters and parameter lists must reside below 16MB.
2. For RACF Version 1 Release 7 and later: If a caller is executing in 24-bit addressing mode, all parameters and parameter lists are assumed to reside below 16MB. If a caller, however, is executing in 31-bit addressing mode and is calling RACF using the RACROUTE macro instruction, RACF will assumes that all parameters and parameter lists may reside above 16MB (that is, that all parameter addresses are true 31-bit addresses).

   All parameter lists generated by the RACROUTE macro are in a format that allows compiled code to be moved above 16MB without recompilation.

   This 31-bit support is available only when RACF is called using RACROUTE, FRACHECK, or RACSTAT. Any caller that uses RACINIT, RACDEF, RACLIST or RACHECK must be in 24-bit addressing mode only. RACF does not support those callers in 31-bit mode.
3. **On z/VM,** the RACROUTE macro described in this section can only be invoked from within the RACF service machine (for example, from an installation exit).

RACSTAT: RACF Status

The RACSTAT macro is used to determine whether RACF is active, and, optionally, determine whether RACF protection is in effect for a given resource class. The RACSTAT macro can also be used to determine whether a resource-class name is defined to RACF.

RACSTAT is a branch-entered service that uses standard linkage conventions.

On z/VM, you can use the RACSTAT macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACSTAT macro from a user's machine.

Note: For RACF release 1.6 and prior releases, only callers in 24-bit addressing mode can issue this macro.

RACSTAT (Standard Form)

The standard form of the RACSTAT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACSTAT.

RACSTAT

␣	One or more blanks must follow RACSTAT.
,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address or register (2) - (12)
,ENTRY=*entry addr*	*entry addr*: A-type address or register (2) - (12)
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6

The parameters are explained as follows:

,CLASS='*class name*'
,CLASS=*class name addr*
specifies the class name for which RACF authorization checking is performed. The name can be explicitly defined on the macro by enclosing the name in quotes. If specified, the address must point to an 8-byte field containing the class name, left-justified and padded with blanks if necessary. If CLASS= is omitted, the status of RACF is returned.

The class name specified must be a general resource defined to RACF in the class-descriptor table. For information on the IBM-supplied classes, see "IBM-Supplied Class Descriptor Table Entries" in *z/VM: RACF Security Server Macros and Interfaces*.

Note: The classes DATASET, USER, and GROUP are not in the class-descriptor table.

,ENTRY=*entry addr*
specifies the address of a 4-byte area that is set to the address of the specified class in the class-descriptor table. This operand is ignored when the CLASS= operand is omitted.

,RELEASE=*number*
specifies the RACF release level of the parameter list to be generated by this macro.

Certain parameters can be specified only with particular releases. If you specify a parameter with an incompatible release level, the parameter is not accepted by the macro processing. An error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 11 on page 274. When you specify the RELEASE keyword, checking is done at assembly time.

The default is RELEASE=1.6.

Parameters for RELEASE=1.6 through 1.8.1

The RELEASE values for which a specific parameter is valid are marked with an 'X'.

Table 11. RACSTAT Parameters for RELEASE=1.6 through 1.8.1

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
CLASS=	X	X	X
ENTRY=	X	X	X

Table 11. RACSTAT Parameters for RELEASE=1.6 through 1.8.1 (continued)

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
RELEASE=	X	X	X

Return Codes

If the return codes and reason codes you are receiving are not discussed in this macro, refer to "Return Codes and Reason Codes" on page 140.

When control is returned, register 15 contains one of the following return codes:

Hexadecimal Code
Meaning

00
RACF is active and, if CLASS= was specified, the class is active.

04
RACF is active; the class is inactive.

08
RACF is active; the class is not defined to RACF.

0C
RACF is inactive and, if CLASS= was specified, the class is active.

10
RACF is inactive; the class is inactive.

14
RACF is inactive; the class is not defined to RACF.

18
RACF is not installed or an insufficient level of RACF is installed.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACSTAT macro; however, the list form of the macro does not have the proper RELEASE parameter. Macro processing terminates.

Note: The class-descriptor entry for the specified class is returned to the caller (in the 4-byte area addressed by the entry address), for return codes 00, 04, 0C, and 10.

Example 1

Operation: Determine whether the DASDVOL class is active and retrieve the address of its class descriptor. A fullword, CDADDR, contains the class-descriptor address.

```
RACSTAT CLASS='DASDVOL',ENTRY=CDADDR
```

RACSTAT (List Form)

The list form of the RACSTAT macro instruction is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACSTAT	One or more blanks must precede RACSTAT.

␣	One or more blanks must follow RACSTAT.
,CLASS='*class name*'	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address.
,ENTRY=*entry addr*	*entry addr*: A-type address.
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
MF=L	

The parameters are explained under the standard form of the RACSTAT macro with the following exception:

MF=L
specifies the list form of the RACSTAT macro.

RACSTAT (Execute Form)

The execute form of the RACSTAT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACSTAT	One or more blanks must precede RACSTAT.
␣	One or more blanks must follow RACSTAT.
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12).
,ENTRY=*entry addr*	*entry addr*: Rx-type address or register (2) - (12).
,RELEASE=*number* ,RELEASE=(,CHECK) ,RELEASE=*(number,*CHECK)	*number*: 1.8.1, 1.8, 1.7, 1.6 **Default:** RELEASE=1.6
MF=(E,*ctrl addr*)	*ctrl addr*: Rx-type address or register (1) - (12).

The parameters are explained under the standard form of the RACSTAT macro with the following exceptions:

MF=(E,*ctrl addr*)
specifies the execute form of the RACSTAT macro, using a remote, control-program parameter list.

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number*,CHECK)
specifies the RACF release level of the parameter list to be generated by this macro.

Certain parameters can be specified only with particular releases. If you specify a parameter with an incompatible release level, the parameter is not accepted by the macro processing. An error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 11 on page 274. When you specify the RELEASE keyword, checking is done at assembly time.

Compatibility between the list and execute forms of the RACSTAT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

RACXTRT Macro

Note: On z/VM, you can use the RACXTRT macro only in the RACF service machine (for example, from an installation exit). You cannot use the RACXTRT macro from a user's machine.

The RACXTRT macro is used to retrieve or replace certain specified fields from a RACF profile or to encode certain clear-text (readable) data.

Note:

1. Encoding, replacement, and extraction are mutually exclusive.
2. Only callers in 24-bit addressing mode can issue this macro. Callers executing in 31-bit addressing mode who want to use the RACXTRT function can code the RACROUTE macro.

The following RACXTRT functions are programming interfaces:

- Retrieving or updating fields in any other product segment in the user, group and resource profiles.
- Retrieving or updating the following installation-reserved fields:
  - USERDATA
  - USRCNT
  - USRDATA
  - USRFLG
  - USRNM
- Retrieving the current or a specified user's default group or password.

The following RACXTRT function is part of the programming interface but is **not** recommended for use because no additional enhancements will be made to this macro.

- Retrieving or updating fields in the base segment of a user, resource, or group profile.

The standard form of the RACXTRT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣	One or more blanks must precede RACXTRT.
RACXTRT	
␣	One or more blanks must follow RACXTRT.

TYPE=EXTRACT
TYPE=EXTRACTN
TYPE=REPLACE
TYPE=ENCRYPT

,ENTITY=*profile name addr* — *profile name addr*: A-type address, or register (2) - (12)

,RELEASE=*number* — *number*: 1.6, 1.7, 1.8, or 1.8.1 **Default:** RELEASE=1.6

,ACEE=*acee-address* — *acee*: A-type address, or register (2) - (12)

,VOLSER=*volser-address* — *vol address*: A-type address, or register (2) - (12)

,GENERIC=ASIS
,GENERIC=YES — **Default:** ASIS

,FLDACC=YES
,FLDACC=NO — **Default:** NO

If TYPE=EXTRACT or EXTRACTN is specified:

,SUBPOOL=*subpool number* — *subpool number*: Decimal digit, 0-255
Default: SUBPOOL=229

,DERIVE=*'YES'* — See explanation of keyword.
Default: Normal processing

,CLASS=*'class name'* — *class name*: 1- to 8-character class name

,CLASS=*class name addr* — *class name addr*: A-type address or register (2) - (12)
Default: 'USER'

,SEGMENT=*'segment name'* — *segment name*: 1- to 8-character name

,SEGMENT=*segment name addr* — *segment name addr*: A-type address or register (2) - (12)

,FIELDS=*field addr*	*field addr*: A-type address or register (2) - (12)

If TYPE=REPLACE is specified:

,CLASS=*'class name'*	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: A-type address or Register (2) - (12) **Default**: 'USER'
,SEGMENT=*'segment name'*	*segment name*: 1- to 8-character name
,SEGMENT=*segment name addr*	*segment name addr*: A-type address or register (2) - (12)
,FIELDS=*field addr*	*field addr*: A-type address or register (2) - (12)
,SEGDATA=*segment data addr*	*segment data addr*: A-type address or register (2) - (12)

If TYPE=ENCRYPT is specified:

,ENCRYPT=(*data address,DES*)	*data address*: A-type address or register (2) - (12)
,ENCRYPT=(*data address,HASH*)	
,ENCRYPT=(*data address,INST*)	

Note: If TYPE=ENCRYPT is specified, the only other allowable parameters are ENTITY, RELEASE, ENCRYPT, with ENCRYPT being required.

The parameters are explained as follows:

TYPE=EXTRACT
specifies the function to be performed by the extract function routine.

With Release 1.8 and later, RACXTRT can provide additional function: it can extract information from any field in any profile. See Appendix B, "RACF Database Templates," on page 293 for a definition of the type and name of each field in each profile. If you specify EXTRACT, the macro extracts information from the profile determined by the ENTITY and CLASS keywords. Specifically, RACF extracts the fields specified in the FIELDS keyword from the segment specified by the SEGMENT keyword. If you do not specify ENTITY, RACF retrieves the desired information from the current user's profile.

To use TYPE=EXTRACT to extract field information from a profile, you must specify Release=1.8 or later.

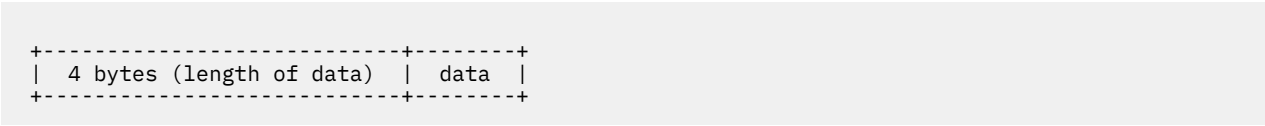
Note: If you specify TYPE=EXTRACT, do not specify ENCRYPT.

Upon return, register 1 contains the address of a result area that begins with a fullword containing the area's subpool number and length. It is your responsibility to issue a FREEMAIN to release the area after you are through using it.

The fields in the result area are in the order below:

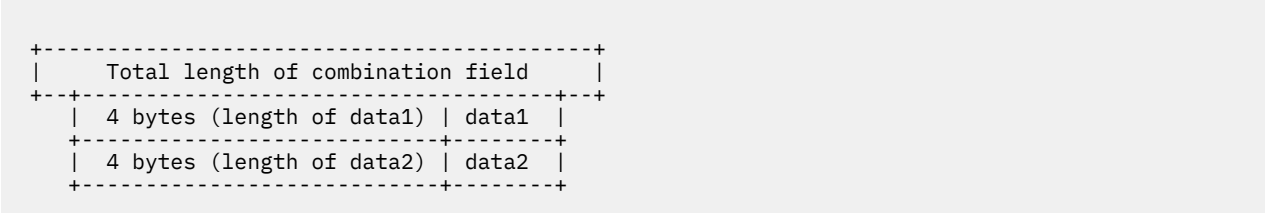
Offset (Dec)	Data	Length (Dec)
0	Subpool of area	1
1	Length of area	3
4	Offset to start of optional field to contain segment data	2
6	Flag	1
7	Reserved	17
24	Specified or current user's user ID, if CLASS=USER	8
32	Specified user's default connect group or current user's current connect group, if CLASS=USER	8

In general, RACF returns field data in the order it was specified, with a 4-byte length field preceding each profile field. For example, if you are extracting a single field, you receive a 4-byte length field that contains the length of the field that follows. If the requested field is a variable length field, there is no additional length byte.



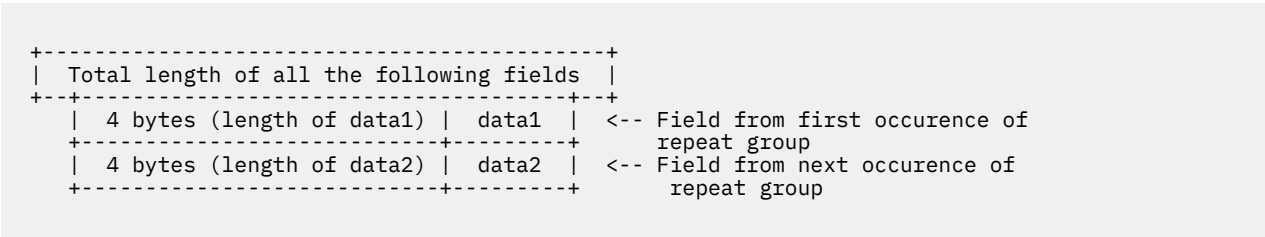
If you are extracting a combination field (representing one or more fields), you receive:

- A 4-byte length field that contains the combined length of all the fields that follow
- A combination field made up of 4-byte length fields followed by their respective individual data fields.



If you are extracting a single field within a repeat group, you receive:

- A 4-byte length field that contains the combined length of all the fields that follow.
- A 4-byte length field that indicates the length of the specified field in the first occurrence of the repeat group. This is followed by a 4-byte length field that indicates the length of the specified field in the second occurrence of the repeat group. The pattern repeats until all the occurrences of the repeat group are accounted for.



If you are extracting a combination field (representing one or more fields) within a repeat group, you receive:

- A 4-byte length field that contains the combined length of all the fields that follow.
- A combination field consisting of a 4-byte length field indicating the length of the individual data field that follows it, followed by the next 4-byte length field indicating the length of the next individual data

field. The pattern repeats until all the individual fields that make up the combination field are accounted for. At the next occurrence of the repeat group the pattern begins again.

```
+-----------------------------------------+
|  Total length of all the occurences of the |
|   combination field in the repeat greoup   |
+--+--------------------------------------+--+
   |  4 bytes (length of data1) |  data1  |
   +----------------------------+---------+  <-- Combination field from first
   |  4 bytes (length of data2) |  data2  |      occurence of repeat group
   +----------------------------+---------+
   |  4 bytes (length of data1) |  data1  |
   +----------------------------+---------+  <-- Combination field from next
   |  4 bytes (length of data2) |  data2  |      occurence of repeat group
   +----------------------------+---------+
```

Specifying the name of a repeat-group count field retrieves only the 4-byte length followed by the 4-byte repeat group count.

When a field to be extracted is empty, the following results:

- For fixed-length fields, RACF returns the default as specified by the template definitions. The default for flag fields is X'00'. The default for fixed-length fields in the base segment of the profile in binary ones. The default for fixed-length fields in other segments is binary zeros.
- For variable-length fields, RACF returns a length of zero and no data.

  If CLASS=USER, when you specify EXTRACT, the macro extracts the user ID, connect group and, optionally, the encoded password from the user profile.

TYPE=EXTRACTN
specifies the function to be performed by the EXTRACT function routine.

Note: If you specify TYPE=EXTRACTN, do not specify ENCRYPT=.

Upon return, register 1 contains the address of a result area that begins with a fullword containing the area's subpool number and length. To see the format of the result area, see the explanation of TYPE=EXTRACT, above. At offset 6 in the result area, there is a flag. If the flag has a X'80', the name returned is generic.

If you specify EXTRACTN, the macro extracts information from the profile that follows the profile determined by the ENTITY and CLASS keywords. From that next profile, RACF extracts the fields specified in the FIELDS keyword from the segment specified by the SEGMENT keyword. In addition, RACF returns the name of the profile from which it extracted the data.

TYPE=REPLACE
specifies the function to be performed by the EXTRACT function routine.

Note: If you specify TYPE=REPLACE, do not specify ENCRYPT=.

Use of the REPLACE option to update a profile requires a thorough knowledge of the interrelationships of fields within a profile and of the potential relationships between profiles. For instance, if you use RACXTRT to update a password, you should also update the password change date and password-history information.

If you specify TYPE=REPLACE, RACF takes the information in the fields specified in the FIELDS parameter and pointed to by SEGDATA, and places that information in the designated SEGMENT. (The SEGMENT is within the profile determined by the ENTITY and CLASS keywords.) If you specify TYPE=REPLACE, you must specify FIELDS, SEGDATA=, and RELEASE=1.8 or later. If you want to replace a segment other than the base segment, you must specify the SEGMENT keyword with the segment you want. If you do not specify SEGMENT, the segment defaults to the base segment.

With 1.8 and later, if you want to create a TSO segment, you can do so by specifying the RACXTRT macro in the following way:

```
TYPE=REPLACE  SEGMENT=TSO
```

,SUBPOOL=*subpool number*
specifies the storage subpool from which the extract-function routine obtains an area needed for the extraction. If this parameter is not specified, it defaults to 229.

Note: Care should be taken in selecting a subpool. Selecting a fetch-protected subpool or subpool 0 may result in programs being unable to access or free retrieved data.

,DERIVE=YES
specifies that the desired field will be obtained from the DFP segment of the appropriate profile. To specify DERIVE, you must also specify RELEASE=1.8.1.

DERIVE requests are limited to the DFP segment of the data-set and user profiles. The following is an explanation of the DERIVE processing for both DATASET and USER requests.

- DATASET

  Specifying the DERIVE=YES keyword with CLASS=DATASET and FIELDS=RESOWNER causes RACF to perform additional processing, other than simply extracting the data-set resource owner from the data-set profile.

  DFP uses this retrieved information for authority checking when allocating a new data set.

  To process the request, RACF first attempts to extract the RESOWNER field from the DATASET profile specified by the ENTITY keyword. If the profile exists and the RESOWNER field contains data, RACF checks to see whether that data is the user ID of a USER or GROUP currently defined to RACF. If so, RACF returns that user ID along with a reason code that indicates whether the user ID is that of a USER or GROUP.

  If RACF does not find a profile that matches the data-set name specified by the ENTITY keyword, RACF attempts to locate the generic data-set profile that protects that data-set name.

  If it finds the generic profile, and the RESOWNER field contains data, RACF checks to see whether that data is the user ID of a USER or GROUP currently defined to RACF. If so, RACF returns that user ID along with a reason code that indicates whether the user ID is that of a USER or GROUP.

  If RACF does not find a generic profile or the retrieved data is neither a USER nor a GROUP, RACF returns the high-level qualifier from the name specified on the ENTITY keyword, along with a reason code that indicates whether that high-level qualifier matches a defined USER or GROUP, or neither.

  You specify a DERIVE request for RESOWNER as follows:

```
RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT,
ENTITY=DSNAME,
VOLSER=MYDASD,
CLASS='DATASET',
FIELDS=RESFLDS,
SEGMENT='DFP',
DERIVE=YES,
RELEASE=1.8.1
.........
DSNAME   DC CL44'USER1.DATASET'
MYDASD   DC CL6'DASD1'
RESFLDS  DC A(1)
         DC CL8'RESOWNER'
```

  Note: You must specify all the keywords in the example, for the DERIVE request to work.

- User

  The purpose of specifying the DERIVE=YES keyword with CLASS=USER is to obtain the desired DFP-field information (STORCLAS or MGMTCLAS) from the profile of the user. If the user's profile does not contain the desired DFP fields, RACF goes to the user's default group and attempts to obtain the information for the remaining fields from the GROUP profile (the remaining fields being those that do not contain information in the USER profile).

  You specify a DERIVE request for information from a USER profile as follows:

```
RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT,
ENTITY=USER01,
CLASS='USER',
```

```
FIELDS=STRFLDS,
SEGMENT='DFP',
DERIVE=YES,
RELEASE=1.8.1
.........
USER01   DC CL8'USER01'
STRFLDS  DC A(1)
         DC CL8'STORCLAS'
```

RACF processes the DERIVE keyword only if it is specified with the DATASET or USER class. In addition, for DERIVE processing to occur, SEGMENT=DFP and RELEASE=1.8.1 must also be specified.

,FIELDS=*address*

Specifies the address of a variable-length list. The first field is a 4-byte field that contains the number of profile-field names in the list that follows. Each profile-field name is 8 bytes long, left-justified, and padded to the right with blanks. The allowable field names for each type of profile are in the template listings in *z/VM: RACF Security Server System Programmer's Guide*. To see how to specify the FIELDS keyword, see the TYPE=REPLACE example that follows.

- If you specify Release=1.6 or later, or allow the keyword to default, the following options exist:
  - The only acceptable value of the count field is 1.
  - The only acceptable field name is PASSWORD. Use this parameter when you want to extract the user's encoded password in addition to the user ID and connect group. RACF returns the encoded password in the result area at an offset from the start of the area specified by the halfword at offset 4. (See the result area under TYPE=EXTRACT.)
- If you specify Release=1.8 or later, the following options exist:
  - The count field can contain numbers from 1 through 255.
  - The field names can be any of the field names in the the template listings.

If you specify TYPE=EXTRACT or EXTRACTN, RACF retrieves the contents of the named fields from the RACF profile indicated by the CLASS= and ENTITY= parameters, and returns the contents in the result area. (See result area explained under the EXTRACT keyword.)

With Release 1.8, you can specify TYPE=REPLACE. RACF replaces or creates the indicated fields in the profile specified on the CLASS and ENTITY keywords with the data pointed to by the SEGDATA keyword.

Note:

1. Do not replace a repeat group count field. Doing so causes unpredictable results.
2. You cannot replace an entire repeat group, a single occurrence of a repeat group, or a single existing field in a repeat group. If you attempt to do so, RACF adds the data to the existing repeat group or groups.

   The only things you can do is retrieve all occurrences of specified fields within a repeat group or add a new occurrence of a repeat group.
3. If you add occurrences of a repeat group, RACF places those additions at the beginning (front) of the repeat group.

The following example of TYPE=REPLACE replaces fields in the base segment. It shows one way to code the macro and the declarations necessary to make the macro work.

```
       RACXTRT  TYPE=REPLACE,
                CLASS='USER',
                ENTITY=USERID,
                FIELDS=FLDLIST,
                SEGDATA=SEGDLIST,
                SEGMENT=BASE

        ..............................

        USERID   DC  CL8,'BILL'
        FLDLIST  DC  A(3)
                 DC  CL8'AUTHOR'
                 DC  CL8'DFLTGRP'
```

```
                DC  CL8'NAME'
        SEGDLIST DC  AL4(6),CL6'JSMITH'
                 DC  AL4(8),CL8'SECURITY'
                 DC  AL4(11),CL11'BILL THOMAS'
        BASE     DC  CL8'BASE'
```

When the replacement action takes place, the following occurs:

- JSMITH is placed in the AUTHOR field in the profile.
- SECURITY is placed in the DFLTGRP field in the profile.
- BILL THOMAS is placed in the NAME field in the profile.

The following example of TYPE=EXTRACT retrieves the universal access from a fully qualified generic data-set profile. The information is retrieved in a work area created in SUBPOOL 1.

```
   RACXTRT    TYPE=EXTRACT,
              CLASS='DATASET',
              ENTITY=DSN,
              FIELDS=FLDS,
              GENERIC=YES,
              SUBPOOL=1
              RELEASE=1.8,
              SEGMENT='TSO'
  ..................
  DSN  DC  CL44'SYS1.LINKLIB'
  FLDS DC  A(1)
       DC  CL8 'UACC'
```

TYPE=ENCRYPT
specifies the function to be performed by the extract-function routine.

If TYPE=ENCRYPT is specified, the operation performed is data encoding. The ENCRYPT keyword specifies the data to be encoded and the encoding method used. The first eight bytes of the area pointed to by the ENTITY operand are used by the data encryption standard (DES) encoding routine. If ENTITY is not specified, the user ID from the current ACEE is used instead. If TYPE=ENCRYPT is specified, no work area is returned.

,ENCRYPT=(*data address*,DES)
,ENCRYPT=(*data address*,HASH)
,ENCRYPT=(*data address*,INST)
specifies the data to be authenticated, and a method of authentication. The address points to a 1-byte length field followed by 1 to 255 bytes of clear-text data to be used as the user-authentication key. The second subparameter specifies the authentication method: the RACF data encryption standard algorithm, the RACF hashing algorithm, or whatever scheme the installation uses (INST value). Upon return to the macro issuer, the first subparameter contains the address of an area that contains a 1-byte length followed by the encoded version of the data. Neither the address itself nor the length is changed.

Note: When the DES algorithm is used, RACF actually encrypts the data pointed to by the ENTITY profile or by the user ID, using the data as the encryption key. Data is one-way encrypted, that is, no facility is provided to recover the data in readable form. If HASH is specified, the RACF hashing algorithm is used and data is masked instead of encrypted.

,ENTITY=*resource name address*
specifies the address of an area containing the resource name. The resource name is a 44-byte DASD data-set name for CLASS='DATASET', an 8-byte area containing the user ID for CLASS='USER', an 8-byte area containing the group ID for CLASS='GROUP', or a 17-byte area for CLASS='CONNECT'. The length of all other resource names is determined from the class-descriptor table. The name must be left-justified in the field and padded with blanks. For CLASS='USER', the user ID from the current ACEE or the ACEE specified for ACEE= will be used if ENTITY= is not specified.

,RELEASE=1.6|1.7|1.8|1.8.1
specifies the RACF release level of the parameter list to be generated by this macro.

To use the parameters associated with a release, you must specify the release number of that release or a later release number. If you specify an earlier release level, the parameter is not accepted by

macro processing, and an error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see Table 12 on page 286.

The default is RELEASE=1.6.

When you specify the RELEASE keyword, checking is done at assembly time. Execution-time validation of the compatibility between the list and execute forms of the RACXTRT macro can be done by specifying the CHECK subparameter on the execute form of the macro.

,ACEE=*acee address*
specifies an alternate ACEE for RACF to use rather than the current ACEE. For example, if the ENTITY parameter has not been specified, RACF refers to the ACEE during extract processing of user data. If you want to use the ACEE parameter, you must specify RELEASE=1.8 or later.

,VOLSER=*vol-address* (valid only with ,CLASS='DATASET')
specifies the volume serial as follows:

- For VSAM DASD data sets and tape data sets, specifies the volume serial number of the catalog controlling the data set.
- For non-VSAM DASD data sets and tape data sets, specifies the volume serial number of the volume on which the data set resides.

The field pointed to by the *vol-address* variable contains the volume serial number. If necessary, you must pad it to the right with blanks so it contain six characters.

If you specify VOLSER, you must specify RELEASE=1.8 or later.

,GENERIC=ASIS|YES
When CLASS is DATASET, specifies whether RACF is to treat the entity name as a generic profile name.

- If you specify GENERIC=YES, RACF considers the entity name a generic profile name, even if it does not contain any of the generic characters (an asterisk or a percent sign).
- If you specify GENERIC=ASIS, RACF considers the entity name a generic only if it contains one or both of the generic characters.

If you specify GENERIC, you must specify RELEASE=1.8 or later.

,FLDACC=NO|YES
specifies whether field-level access checking should be performed. If you specify FLDACC=YES, the RACF database manager checks to see that the user running your program has the authority to extract or modify the fields that are specified in the RACXTRT macro.

Note:

1. For field-level access checking to occur, you must specify RELEASE=1.8 or later when you code the macro. In addition, before the program executes, the security administrator must activate the FIELD class. If you code FLDACC=YES and the field class is not active, the request fails with a return code 8 reason code 4.
2. In addition, the security administrator must issue the RDEFINE and PERMIT commands to designate those users who will have the authority to access the fields designated in the RACXTRT macro.
3. If you specify FLDACC=NO or omit the parameter, the manager ignores field-level access checking.

,CLASS='*class name*'
,CLASS=*class name addr*
specifies the class the entity is in. The class name can be USER, GROUP, CONNECT, DATASET, or any general-resource class defined in the class-descriptor table. If you specify CLASS, you must specify RELEASE=1.8 or later.

,SEGMENT='*segment name*'
,SEGMENT=*segment name address*
specifies the RACF profile segment that RACF is to update or from which it is to extract data. If you specify SEGMENT, you must also specify the CLASS and FIELDS keywords, and RELEASE=1.8 or a

later release number. If you allow the SEGMENT parameter to default, RACF assumes that you want to extract information from the base segment.

,SEGDATA=*segment data addr*
specifies the address of a list of data items to be placed in the fields named by the FIELDS= parameter. You use the SEGDATA parameter when you specify TYPE=REPLACE. If you specify SEGDATA, you must also specify CLASS, FIELDS, and RELEASE=1.8 or a later release number. The stored data is paired in the following format:

- A 4-byte length field that contains the length of the data field that follows
- A data field of variable length.

Each length field is followed immediately by a data field until you reach the end of the replacement data. The count field, which is pointed to by the first field in the FIELDS parameter, contains the total number of length-data pairs.

Parameters for RELEASE=1.6 through 1.8.1

The RELEASE values for which a specific parameter is valid are marked with an 'X'.

Table 12. RACXTRT Parameters for RELEASE=1.6 through 1.8.1

Parameter	RELEASE=1.6	RELEASE=1.7	RELEASE=1.8 or 1.8.1
ACEE=			X
CLASS=			X
DERIVE=YES			X
ENCRYPT=	X	X	X
ENTITY=	X	X	X
EXTRACT=	X	X	X
EXTRACTN=			X
FLDACC=			X
FIELDS=	X	X	X
GENERIC=			X
RELEASE=	X	X	X
REPLACE=			X
SEGDATA=			X
SEGMENT=			X
SUBPOOL=	X	X	X
TYPE=	X	X	X
VOLSER=			X

Return Codes and Reason Codes

If the return codes and reason codes you are receiving are not discussed in this macro, refer to "Return Codes and Reason Codes" on page 106.

When control is returned, register 15 contains one of the following return codes, and register 0 may contain a reason code.

Hexadecimal Code
Meaning

00
The extraction or encoding completed successfully.

Reason Code —
For Derive Requests

0
Some of the values are derived from the USER profile, and some may be derived from the GROUP profile.

4
High-level qualifier returned as RESOWNER. It matched a valid USER.

8
DFP data returned from an EXTRACT request from USER profile was actually from the user's default connect group.

0C
High-level qualifier returned as RESOWNER. It matched a valid GROUP.

24
RESOWNER field matched a valid USER.

28
RESOWNER field matched a valid GROUP.

2C
At least one, but not all, of the fields requested failed to be retrieved because of field level access checking.

04
An ESTAE environment was not able to be established, or if register 0 contains a reason code of 1, neither EXTRACT nor ENCRYPT was specified for TYPE=.

08
For TYPE=EXTRACT, TYPE=EXTRACTN, or TYPE=REPLACE the profile could not be found. The hexadecimal reason codes are:

0
No profile found.

4
Field-level access checking failed. The field class may not be active.

8
Segment not found.

14
Neither the RESOWNER field nor the high-level qualifier matched a valid USER or GROUP.

C
RACF is inactive.

10
The extract operation failed. Register 0 contains the RACF-manager return code that represents the cause of termination. This return code is not used for the encoding function. The manager return code and reason codes are returned in the low-order and high-order halfwords of register 0.

14
For TYPE=ENCRYPT or TYPE=EXTRACT of user-class data, ENTITY was specified and no ACEE exists, or the ACEE was not for a defined user.

0
No ACEE exists.

4
ACEERACF bit is off.

18
A parameter-list error was encountered. The hexadecimal reason codes are:

4
For TYPE=REPLACE request, FIELDS= was not specified.

8
Invalid type specified.

C
Invalid number of fields.

10
Invalid class name specified.

14
Invalid version in parameter list.

18
Invalid subpool specified.

1C
Invalid parameter length.

20
For TYPE=REPLACE request, SEGDATA= was not specified.

24
Invalid entity name specified.

2C
For TYPE=ENCRYPT request, no user-authentication key was specified.

30
Invalid encoding method.

34
ENTITY= was not specified with TYPE=REPLACE, TYPE=EXTRACTN, or TYPE=EXTRACT with class other than USER.

38
Multiple profiles; no volume specified.

3C
Profile found wrong volume serial number specified.

48
Invalid entity-name length with the ENTITY keyword:

- The specified length is one of the following:
  - Greater than 44 if CLASS=DATASET
  - Greater than 8 if CLASS=USER or GROUP
  - Greater than 17 if CLASS=CONNECT
  - Greater than the maximum for the specified class as defined in the class-descriptor table.
- For TYPE=ENCRYPT request, the specified length is not zero or eight.

50
The entity name contains a blank at the beginning or in the middle of the name.

64
Indicates that the CHECK subparameter of the RELEASE keyword was specified on the execute form of the RACXTRT macro; however, the list form of the macro does not have the proper RELEASE parameter. It also indicates that the TYPE parameters specified on the list and execute forms may not be the same TYPE. Macro processing terminates.

RACXTRT (List Form)

The list form of the RACXTRT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.

␣	One or more blanks must precede RACXTRT.
RACXTRT	
␣	One or more blanks must follow RACXTRT.
TYPE=EXTRACT TYPE=EXTRACTN TYPE=REPLACE TYPE=ENCRYPT	
,ENTITY=*resource name addr*	*resource name addr*: A-type address
,RELEASE=*number*	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6
,ACEE=*acee-address*	*acee*: A-type address
,VOLSER=*volser-address*	*vol address*: A-type address
,GENERIC=ASIS ,GENERIC=YES	**Default:** ASIS
,FLDACC=YES ,FLDACC=NO	**Default:** NO
,MF=L	
If TYPE=EXTRACT or EXTRACTN is specified:	
,SUBPOOL=*subpool number*	**Default:** SUBPOOL=229
,DERIVE=*YES*	See explanation of keyword. **Default:** Normal processing
,CLASS=*'class name'* ,CLASS=*class name addr*	*class name*: 1- to 8-character class name *class name addr*: A-type address **Default**: CLASS='USER'
,SEGMENT=*'segment name'* ,SEGMENT=*segment name addr*	*segment name*: 1- to 8-character name *segment name addr*: A-type address

,FIELDS=*field addr*	*field addr*: A-type address
If TYPE=REPLACE is specified:	
,CLASS='*class name*',	*class name*: 1- to 8-character class name
,CLASS=*class name addr*	*class name addr*: Rx-type address or Register (2) - (12) **Default:** CLASS='USER'
,SEGMENT='*segment name*'	*segment name*: 1- to 8-character name
,SEGMENT=*segment name addr*	*segment name addr*: A-type address
,FIELDS=*field addr*	*field addr*: A-type address
,SEGDATA=*segment data addr*	*segment data addr*: A-type address
If TYPE=ENCRYPT is specified:	
,ENCRYPT=*(data address,DES)* ,ENCRYPT=(*data address,HASH*) ,ENCRYPT=(*data address,INST*)	*data address*: A-type address

The parameters are explained under the standard form of the RACXTRT macro with the following exception:

,MF=L
specifies the list form of the RACXTRT macro.

RACXTRT (Execute Form)

The execute form of the RACXTRT macro is written as follows:

name	*name*: Symbol. Begin *name* in column 1.
␣ RACXTRT	One or more blanks must precede RACXTRT.
␣	One or more blanks must follow RACXTRT.
TYPE=EXTRACT TYPE=ENCRYPT	
,ENTITY=*resource name addr*	*resource name addr*: Rx-type address or register (2)-(12)
,RELEASE=*number* ,RELEASE=(,CHECK)	*number*: 1.8.1, 1.8, 1.7, or 1.6 **Default:** RELEASE=1.6

,RELEASE=*(number,*CHECK)	
,ACEE=*acee-address*	*acee*: Rx-type address or register (2) - (12)
,VOLSER=*volser-address*	*vol address*: Rx-type address or register (2) - (12)
,GENERIC=ASIS ,GENERIC=YES	
,FLDACC=YES ,FLDACC=NO	
,MF=(E,*ctrl addr*)	ctrl addr: Rx-type address, register (1) or register (2) - (12)
If TYPE=EXTRACT or EXTRACTN is specified:	
,SUBPOOL=*subpool number*	*subpool number*: Decimal digit 0-255
,DERIVE=*YES*	See explanation of keyword.
,CLASS=*class name addr*	*class name addr*: Rx-type address or register (2) - (12)
,SEGMENT=*segment name addr*	*segment name addr: Rx-type address or register (2) - (12)*
,FIELDS=*field addr*	*field addr: Rx-type address or register (2) - (12)*
If TYPE=REPLACE is specified:	
,CLASS=*class name addr*	*class name addr: Rx-type address or Register (2) - (12)*
,SEGMENT=*segment name addr*	*segment name addr: Rx-type address or register (2) - (12)*
,FIELDS=*field addr*	*field addr: Rx-type address or register (2) - (12)*
,SEGDATA=*segment data addr*	*segment data addr: Rx-type address or register (2) - (12)*
If TYPE=ENCRYPT is specified:	
,ENCRYPT=(*data address,DES)* ,ENCRYPT=(*data address,HASH)* ,ENCRYPT=(*data address,INST)*	*data address: Rx-type address or register (2) - (12)*

The parameters are explained under the standard form of the RACXTRT macro with the following exceptions:

,RELEASE=*number*
,RELEASE=(,CHECK)
,RELEASE=(*number,CHECK)*
specifies the RACF release level of the parameter list to be generated by this macro.

Certain parameters can be specified only with particular releases. If you specify a parameter with an incompatible release level, the parameter is not accepted by the macro processing. An error message is issued at assembly time. For the parameters that are valid for RELEASE=1.6 and later, see “Parameters for RELEASE=1.6 through 1.8.1” on page 286.

When you specify the RELEASE keyword, checking is done at assembly time. Compatibility between the list and execute forms of the RACXTRT macro will be validated at execution time if you specify the CHECK subparameter on the execute form of the macro.

The size of the list form expansion must be large enough to accommodate all parameters defined by the RELEASE keyword on the execute form of the macro. Otherwise, when CHECK processing is requested, the execute form of the macro is not done, and a return code of X'64' is returned.

The default is RELEASE=1.6.

,MF=(E,*ctrl addr)*
specifies the execute form of the RACXTRT macro, using a remote, control-program parameter list.

Appendix B. RACF Database Templates

Note: These templates include fields from z/OS V1R8. Not all of these segments and fields can be created by RACF on z/VM.

Included in this appendix are the following templates:

1. GROUP
2. USER
3. CONNECT
4. DATA SET
5. GENERAL
6. RESERVED

Attention
Do not modify the RACF database templates (CSECT IRRTEMP2). Such modification is not supported by IBM and might result in damage to your RACF database or other unpredictable results.

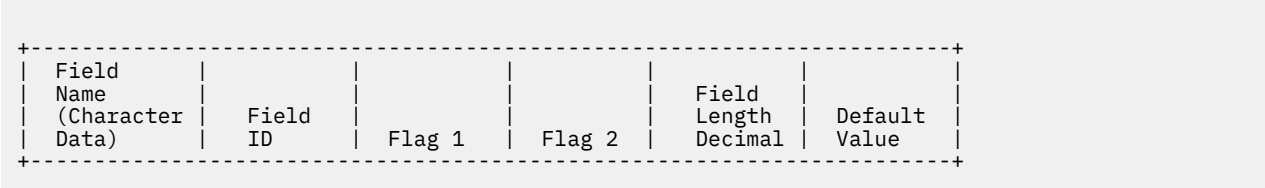
Note: The first field in a segment of a template cannot be retrieved or updated. This field has a Field ID of 001 and is usually described in the **'Field Being Described'** column as 'Start of Segment Fields'.

Format of Field Definitions (RACF Database)

The restructured RACF database templates contain a 31-byte definition for each field in the profile.

Note: The field reference number (Field ID) is now part of the templates.

Each field definition contains information about the field in the following format:



Field Name		**Character data**
Field ID		**Reference Number**
Flag 1 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.
	Bit 6:	The field is sorted.
	Bit 7:	The field is a statistical field. A value is always stored for this field, even when it is equal to the defined null value for the field.

© Copyright IBM Corp. 1990, 2022

Field Name		**Character data**
Field ID		**Reference Number**
Flag 1 field		**The bits have the following meanings when they are turned on:**
Flag 2		The bits have the following meanings when they are turned on:
	Bit 0:	Changes to this field affect security and cause ACEEs to be purged from VLF.
	Bit 1:	The field is padded on the left with binary zeros when values less than the field length are retrieved.
	Bit 2:	This field represents a 3-byte date field.
	Bit 3:	This field is an Application Identity Mapping alias name.
	Bit 4:	This field is not to be unloaded by the Database Unload utility (IRRDBU00).
	Bit 5:	The alias name in this field is EBCDIC.
	Bit 6-7:	Reserved for IBM use.
Field Length		Field length on return from ICHEINTY or RACROUTE REQUEST=EXTRACT (0 is variable length).
Default Value		Field default. If the field is not present in the profile, this byte is propagated throughout the returned field as the default value.
Type		Data type of each field. In this column, character is represented as 'Char', integer is represented as 'Int', and binary is represented as 'Bin'. 'Date' and 'Time' are also possible data types. The type of a combination field that represents a single field is the same as that single field. There is no "type" associated with a combination field which represents multiple fields.

Repeat Groups on the RACF Database

A repeat group consists of one or more sequential fields within a profile that are able to be repeated within that profile. 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.

Field Length

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 0. In both cases, the actual field length is contained in the physical data mapped by the field definition.

Data field types

RACF stores information in the RACF database in many different formats. This section identifies the major data types that RACF stores. Exceptions and additional detail can be found in the description of each specific field within the templates.

Date fields

The format of the 3-byte date fields is *yydddF*, which represents a packed decimal number in which *y* represents year, *d* represents day, and *F* represents the sign. Examples of RACF date values are X'98111C' and X'94099D'.

The format of the 4-byte date fields should be *yyyymmdd*, which represents a packed decimal number in which *y* represents year, *m* represents month, and *d* represents day. Examples of RACF date values are X'19980421' and X'19940409'.

RACF might use any of the following values for null dates: X'FFFFFF', X'00000D', X'00000C', and X'000000' for 3-byte addresses, and X'FFFFFFFF', X'0000000D', X'0000000C', and X'00000000' for 4-byte addresses. However, you should always set null dates to either X'00000F' for 3-byte addresses and X'0000000F' for 4-byte addresses.

Time fields
The format for the 4-byte time fields are *hhmmsstc* where *h* represents hours, *m* represents minutes, *s* represents seconds, *t* represents tenths of seconds, and *c* represents hundredths of seconds. There is no sign byte.

Integer fields
Integers are stored as unsigned binary values. These values can be 1, 2, or 4 bytes in length.

Character fields
Character fields are padded with blanks to the right.

Combination Fields on the RACF Database

The database templates also contain definitions called *combination fields.*

Combination fields do not describe a field of a profile. They contain the field numbers that identify the respective field definitions. You can use the combination field to access multiple fields with one ICHEACTN or RACROUTE REQUEST=EXTRACT macro.

In addition, you can use the combination field to provide aliases for individual fields.

The format of a combination field definition is different from a non-combination definition. Its format is as follows:

Field Name	Character data.
Field ID	Reference number.
Flag 1	The hex representation of the flag bits for this field. For combination fields, bit 1 is on. For a continuation of combination fields, bit 4 is also on.
Flag 2	The hex representation of the flag bits for this field. For combination fields, all bits are off.
Combination IDs	If nonzero, combination IDs represent the position of a non-combination field within the template segment. Up to 5 numbers are allowed.
Comments	Comment field.

Determining Space Requirements for the Profiles

The formula for calculating the space required for each segment (Base RACF information, TSO, DFP, and so on) of each profile in the restructured RACF database is as follows:

```
P = 20 + L + F1 + F4 + R
```

Where:

P	=	The number of bytes required for a profile segment
L	=	The number of bytes in the profile name

F1 = The sum of the lengths of all fields that contain data and have a length of 1 to 127 bytes, plus 2 bytes for every field counted.

For example, if a segment contains 3 non-null fields of length 8, `F1 = (3 * 8) + (3 * 2) = 24 + 6 = 30`.

F4 = The sum of the lengths of all fields that contain data and have a length of 128 to 2**31 bytes, plus 5 bytes for every field counted.

For example, if a segment contains a non-null field 150 bytes long and a non-null field 255 bytes long, `F4 = 150 + 255 + (2 * 5) = 150 + 255 + 10 = 415`

R = The sum of the lengths of all repeat groups. If a repeat group has no occurrences, then it has a length of 0 bytes. If a repeat group has 1 or more occurrences, then the length of each repeat group is calculated as follows:

`9 + N + G1 + G4`

N = The number of occurrences of the group

G1 = The sum of the lengths of all fields in the group, which have a length of 1 to 127 bytes, plus 1 byte for every field counted. If a field has a length of zero, it will still take up 1 byte in the profile.

G4 = The sum of the lengths of all fields in the group, which have a length of 128 to 2**31 bytes, plus 4 bytes for every field counted.

For example, consider a group with two occurrences. Each occurrence contains an 8-byte field and a variable length field. In the first occurrence, the variable length field is 30 bytes and in second occurrence, it is 200 bytes. The length of the group is:

`9 + 2 + G1 + G4`

G1 is `(8 + 1) + (30 + 1)` from the first occurrence and `(8 + 1)` from the second, for a total of 49 bytes. G4 is `(200 + 4)` from the second occurrence, or 204 bytes. So, the length of the group is `9 + 2 + 49 + 204`, or 264 bytes.

Note: For each repeat group the sum of G1 and G4 may not exceed 65535 bytes. For example, this would translate into a maximum of 8191 group connections per user. As another example, this would translate into a maximum of 5957 users connected to a group.

When calculating F1 and F4, remember that statistical fields (Flag1/bit 7 on, in the template definition) are always stored in a profile segment, even when the field contains a null value. For example, REVOKECT

will always add 3 bytes to the length of a USER profile Base segment, regardless of whether it contains a zero value or some other value. Other fields will only exist in the segment, if a specific value has been added for that field.

Note: The RACF database space required for a segment is a multiple of the 256-byte slots required to contain the segment. For example, if a USER profile Base segment contains 188 bytes of data, it will still require 256 bytes of space in the RACF database.

Group template for the RACF database

NOT programming interface information

ACSCNT
FIELD
FLDCNT
FLDFLAG
FLDNAME
FLDVALUE
INITCNT

End of NOT programming interface information

Note: Application developers should not depend on being able to use RACROUTE REQUEST=EXTRACT for the BASE segment fields on any security product other than RACF. These products are expected to support only such segments as DFP and TSO.

The contents of the group template are as follows:

Template							**Field being described**
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
The following is the BASE segment of the GROUP template.							
GROUP	001	00	00	00000000	00		
ENTYPE	002	00	00	00000001	01	Int	The number (1) corresponding to group profiles.
VERSION	003	00	00	00000001	01	Int	The version field from the profile. Always X'01'.
SUPGROUP	004	00	80	00000008	FF	Char	The superior group to this group.
AUTHDATE	005	00	20	00000003	FF	Date	The date the group was created.
AUTHOR	006	00	80	00000008	FF	Char	The owner (user ID or group name) of the group.
INITCNT	007	00	00	00000002	FF		Reserved for IBM's use.
UACC	008	20	00	00000001	00	Bin	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 for IBM's use **Note:** This field has a value of X'00', except for the IBM-defined group VSAMDSET, where the value is X'20'.
NOTRMUAC	009	20	00	00000001	00	Bin	If bit 0 is on, the user must be specifically authorized (by the PERMIT command) to use the terminal. If off, RACF uses the terminal's UACC.
INSTDATA	010	00	00	00000000	00	Char	Installation data.
MODELNAM	011	00	00	00000000	00	Char	Data set model profile name. The profile name begins with the second qualifier; the high-level qualifier is not stored.
FLDCNT	012	10	00	00000004	00		Reserved for IBM's use.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
FLDNAME	013	80	00	00000008	00		Reserved for IBM's use.
FLDVALUE	014	80	00	00000000	00		Reserved for IBM's use.
FLDFLAG	015	A0	00	00000001	00		Reserved for IBM's use.
SUBGRPCT	016	10	00	00000004	00	Int	The number of subgroups of the group.
SUBGRPNM	017	80	80	00000008	00	Char	A list of the subgroup names.
ACLCNT	018	10	00	00000004	00	Int	The number of users connected to the group.
USERID	019	80	00	00000008	00	Char	The user ID of each user connected to the group.
USERACS	020	A0	00	00000001	00	Bin	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 for IBM's use
ACSCNT	021	80	00	00000002	00		Reserved for IBM's use.
USRCNT	022	10	00	00000004	00	Int	Reserved for installation's use. See **Note 1.**
USRNM	023	80	00	00000008	00		Reserved for installation's use. See **Note 1.**
USRDATA	024	80	00	00000000	00		Reserved for installation's use. See **Note 1.**
USRFLG	025	A0	00	00000001	00		Reserved for installation's use. See **Note 1.**
UNVFLG	026	20	00	00000001	00	Bin	Identifies the group as having (bit 0 is on) or not having the UNIVERSAL attribute.

Note 1: Intended usage for these fields is to allow the installation to store additional data in this profile. USRNM should have a field name to use as a key to identify each unique occurrence of a row in the repeat group. USRDATA and USRFLG hold the data associated with that name. For more information, see "Example 5: Updating the installation fields" in "Examples of ICHEINTY, ICHETEST, and ICHEACTN Macro Usage" in Appendix B of *z/VM: RACF Security Server Macros and Interfaces* .

Field name	**Field ID**	**Flag 1**	**Flag 2**	**Combination field IDs**					**Type**	
The following are the COMBINATION fields.										
DEFDATE	000	40	00	005	000	000	000	000	Char	Alias for AUTHDATE
CREADATE	000	40	00	005	000	000	000	000	Char	Alias for AUTHDATE
OWNER	000	40	00	006	000	000	000	000	Char	Alias for AUTHOR
FIELD	000	40	00	013	014	015	000	000		FLDNAME, FLDVALUE, and FLDFLAG
ACL	000	40	00	019	020	021	000	000		USERID, USERACS, and ACSCNT
USERDATA	000	40	00	023	024	025	000	000		USERNM, USERDATA, and USERFLG

Template							Field Being Described
Field Name (Character Data)	**Field ID**	**Flag 1**	**Flag 2**	**Field Length Decimal**	**Default Value**	**Type**	
The following is the DFP Segment of the GROUP Template.							
DFP	001	00	00	00000000	00		Start of segment
DATAAPPL	002	00	00	00000000	00	Char	Data Application
DATACLAS	003	00	00	00000000	00	Char	Data Class
MGMTCLAS	004	00	00	00000000	00	Char	Management Class
STORCLAS	005	00	00	00000000	00	Char	Storage Class
The following is the OMVS Segment of the GROUP Template.							
OMVS	001	00	00	00000000	00		Start of segment
GID	002	00	10	00000004	FF	Int	GID
The following is the OVM Segment of the GROUP Template.							
OVM	001	00	00	00000000	00		Start of segment
GID	002	00	00	00000004	FF	Int	GID
The following is the TME Segment of the GROUP Template.							
TME	001	00	00	00000000	00		Start of segment fields
ROLEN	002	10	00	00000004	00	Int	Count of roles
ROLES	003	80	00	00000000	00	Char	Role names

User Template for the Restructured Database

The user template describes the fields of the user profiles in a RACF database.

NOT programming interface information

CATEGORY	FLDVALUE	OLDPWDX	PHRCNT
CONGRPCT	MAGSTRIP	OLDPHREX	PPHENV
CONGRPNM	NUMCTGY	OPWDX	PREVKEY
CURKEY	OLDPHR	OPWDXCT	PREVKEYV
CURKEYV	OLDPHRES	OPWDXGEN	PWDCNT
ENCTYPE	OLDPHRNM	PASSWORD	PWDENV
FIELD	OLDPHRX	PHRASE	PWDGEN
FLDCNT	OLDPHRNX	PHRASEX	PWDX
FLDFLAG	OLDPWD	PHRCNTX	SALT
FLDNAME	OLDPWDNM	PHRGEN	

End of NOT programming interface information

Notes:

1. Application developers should not depend on being able to use RACROUTE REQUEST=EXTRACT for the BASE segment fields on any security product other than RACF. These products are expected to support only such segments as DFP and TSO.
2. PASSWORD and PHRASE are not programming interface fields when KDFAES is the active encryption algorithm.

The contents of the user template (base segment) are as follows:

Template							**Field being described**
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
The following is the BASE segment of the USER template.							
USER	001	00	00	00000000	00		
ENTYPE	002	00	00	00000001	02	Int	The number (2) corresponding to user profiles.
VERSION	003	00	00	00000001	01	Int	The version field from the profile. Always X'01'.
AUTHDATE	004	00	20	00000003	FF	Date	The date the user was defined to RACF.
AUTHOR	005	00	00	00000008	FF	Char	The owner (user ID or group name) of the user profile.
FLAG1	006	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having the ADSP attribute.
FLAG2	007	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having the SPECIAL attribute.
FLAG3	008	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having the OPERATIONS attribute.
FLAG4	009	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having the REVOKE attribute.
FLAG5	010	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having the GRPACC attribute.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
PASSINT	011	00	80	00000001	FF	Int	The interval in days (represented by a number between 1 and 254) that the user's password is in effect. If it is X'FF', the user's password never expires. See the description of the SETR PASSWORD(INTERVAL...)) processing instructions in *z/VM: RACF Security Server Command Language Reference* for more details.
PASSWORD	012	04	80	00000008	FF	Char	The password associated with the user. For masking, the masked password is stored. For DES or KDFAES, the encrypted user ID is stored. If the installation provides its own password authentication, data returned by the ICHDEX01 exit is stored.
PASSDATE	013	00	20	00000003	FF	Date	The date the password was last changed.
PGMRNAME	014	00	00	00000020	FF	Char	The name of the user.
DFLTGRP	015	00	00	00000008	FF	Char	The default group associated with the user. A value of X'FF' indicates that no group was specified.
LJTIME	016	01	00	00000004	FF	Time	The time that the user last entered the system by using RACROUTE REQUEST=VERIFY.
LJDATE	017	01	20	00000003	FF	Date	The date that the user last entered the system by using RACROUTE REQUEST=VERIFY.
INSTDATA	018	00	80	00000000	00	Char	Installation data.
UAUDIT	019	20	80	00000001	00	Bin	Identifies whether all RACROUTE REQUEST=AUTH, RACROUTE REQUEST=DEFINE, (and, if the caller requests logging, RACROUTE REQUEST=FASTAUTH) macros 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. If bit 0 is off, logging might still occur for other reasons, as identified in *z/VM: RACF Security Server Auditor's Guide* .
FLAG6	020	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having the AUDITOR attribute.
FLAG7	021	20	80	00000001	00	Bin	If bit 1 is on, this is a protected user ID, which cannot enter the system by any means requiring a password, password phrase, or MFA. If bit 2 is on, this user can enter the system with a password phrase.
FLAG8	022	20	80	00000001	00	Bin	If bit 0 is on, an operator identification card (OID card) is required when logging on to the system. If bit 1 is on, the user must authenticate with MFA unless a LOGON FALLBACK is permitted. If bit 2 is on, the user is permitted to use LOGON FALLBACK to authenticate with other factors such as password or password phrase
MAGSTRIP	023	04	00	00000000	00	Bin	The operator identification associated with the user from the masked or encrypted OID card data required to authenticate this user, as supplied by a supported 327x (such as 3270 and 3278) OID card reader.
PWDGEN	024	00	00	00000001	FF	Int	Current password generation number.
PWDCNT	025	10	00	00000004	00	Int	Number of old passwords present.
OLDPWDNM	026	80	00	00000001	00	Int	Generation number of previous password.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
OLDPWD	027	84	00	00000008	FF	Char	Previous password. This is an encrypted password value.
REVOKECT	028	01	80	00000001	FF	Int	Count of unsuccessful password attempts. **Note:** You can use ALTER when setting this field, but you cannot use ALTERI.
MODELNAM	029	00	80	00000000	00	Char	Data set model profile name. The profile name begins with the second qualifier; the high-level qualifier is not stored.
SECLEVEL	030	00	80	00000001	FF	Int	The number that corresponds to the user's security level. For more information on security levels, see *z/OS: Security Server RACF Security Administrator's Guide* .
NUMCTGY	031	10	80	00000004	00	Int	Number of security categories.
CATEGORY	032	80	80	00000002	00	Int	A number that corresponds to the security categories to which the user has access.
REVOKEDT	033	00	20	00000000	00	Date	The date the user will be revoked. This field either has length 0, or contains a 3-byte revoke date.
RESUMEDT	034	00	20	00000000	00	Date	The date the user will be resumed. This field either has length 0, or contains a 3-byte resume date.
LOGDAYS	035	20	00	00000001	00	Bin	The days of the week the user cannot log on (Bit 0 of this field equals Sunday, bit 1 equals Monday, and so on).
LOGTIME	036	00	80	00000000	00	Time	The time of the day the user can log on. If present (length of variable field not equal to 0), it is specified as 6 bytes formatted as two 3-byte packed decimal fields, *0ssssC0eeee*C, where *ssss* represents the start time (*hhmm*) from the ALU...WHEN(TIMES(...)) specification and *eeee* represents the end time. For *hhmm*, *hh* represents hours, and *mm* represents minutes.
FLDCNT	037	10	00	00000004	00		Reserved for IBM's use.
FLDNAME	038	80	00	00000008	00		Reserved for IBM's use.
FLDVALUE	039	80	00	00000000	00		Reserved for IBM's use.
FLDFLAG	040	A0	00	00000001	00		Reserved for IBM's use.
CLCNT	041	10	80	00000004	00	Int	The number of classes in which the user is allowed to define profiles.
CLNAME	042	80	80	00000008	00	Char	A class in which the user is allowed to define profiles. (The user has the CLAUTH attribute.) The user can also define profiles in any other classes with POSIT values matching these classes.
CONGRPCT	043	10	80	00000004	00	Int	The number of groups that the user is connected to.
CONGRPNM	044	80	80	00000008	00	Char	A group that the user is connected to.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
USRCNT USRNM USRDATA USRFLG	045 046 047 048	10 80 80 A0	00 80 80 80	00000004 00000008 00000000 00000001	00 00 00 00	Int	Reserved for installation's use. **Note:** Intended usage: For installation to store additional data in this profile. USRNM should have a field name to use as a key to identify each unique occurrence of a row in the repeat group. USRDATA and USRFLG hold the data associated with that name. For more information, see "Example 5: Updating the installation fields" in "Examples of ICHEINTY, ICHETEST, and ICHEACTN Macro Usage" in Appendix B of *z/VM: RACF Security Server Macros and Interfaces* .
SECLABEL	049	00	80	00000008	00	Char	Security label.
CGGRPCT	050	10	80	00000004	00	Int	Number of Connect Group entries. Information from the following CGxxx fields is also available through the logical connect profiles (ICHEINTY with CLASS=CONNECT) in the database. See "Connect Template for the Restructured Database" on page 315 for more details.
CGGRPNM	051	82	80	00000008	00	Char	Connect Group Entry Name.
CGAUTHDA	052	80	A0	00000003	FF	Date	Date the user was connected.
CGAUTHOR	053	80	80	00000008	FF	Char	Owner of connect occurrence.
CGLJTIME	054	81	00	00000004	FF	Time	Time of RACROUTE REQUEST=VERIFY.
CGLJDATE	055	81	20	00000003	FF	Date	Date of RACROUTE REQUEST=VERIFY.
CGUACC	056	A0	80	00000001	00	Bin	Default universal access.
CGINITCT	057	81	00	00000002	FF	Int	Number of RACROUTE REQUEST=VERIFY requests that were successfully processed where the value specified in the CGRPNM field was the current connect group.
CGFLAG1	058	A0	80	00000001	00	Bin	If bit 0 is on, the user has the ADSP attribute in that group.
CGFLAG2	059	A0	80	00000001	00	Bin	If bit 0 is on, the user has the SPECIAL attribute in that group.
CGFLAG3	060	A0	80	00000001	00	Bin	If bit 0 is on, the user has the OPERATIONS attribute in that group.
CGFLAG4	061	A0	80	00000001	00	Bin	If bit 0 is on, the user has the REVOKE attribute in that group.
CGFLAG5	062	A0	80	00000001	00	Bin	If bit 0 is on, the user has the GRPACC attribute in that group.
CGNOTUAC	063	A0	80	00000001	00	Bin	If bit 0 is on, the user must be specifically authorized (by the PERMIT command) to use a terminal. If off, RACF uses the terminal's UACC.
CGGRPAUD	064	A0	80	00000001	00	Bin	If bit 0 is on, the user has the GROUP AUDITOR attribute in that group.
CGREVKDT	065	80	20	00000000	00	Date	The date the user will be revoked. This field either has length 0, or contains a 3-byte revoke date.
CGRESMDT	066	80	20	00000000	00	Date	The date the user will be resumed. This field either has length 0, or contains a 3-byte resume date.
TUCNT	067	10	00	00000002	00	Int	Number of user ID associations.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
TUKEY	068	80	00	00000016	00	Char	Associated node and user ID. **Byte** **Meaning when set** **0–7** The associated node name. **8–15** The associated user ID.
TUDATA	069	80	00	00000000			Associated user ID association data **Byte** **Meaning when set** **0** Version number of the TUDATA entry.
						Bin	**1** Bitstring **0** Specifies the user as having (bit is on) or not having (bit is off) a peer user ID association. **1** Specifies the user as being (bit is on) the manager of a managed user ID association. **2** Specifies the user as being (bit is on) managed by a managed user ID association. **3** An association request for this user is pending (bit is on) on a remote RRSF node. **4** An association request for this user is pending (bit is on) on the local RRSF node. **5** Specifies that password synchronization is in effect (bit is on) for this peer-user ID association. **6** Specifies that the association request for this user was rejected (bit is on). **7** Reserved for IBM's use. **2–20** Reserved for IBM's use.
						Date	**2–24** The date the user ID association was defined. (*yyyymmdd*)
						Time	**25–32** The time the user ID association was defined. For the format of the time, see the TIME macro as documented in *z/OS MVS Programming: Assembler Services Reference, Volume 2 (IARR2V-XCTLX)* .

Template							Field being described
Field name (character data)	Field ID	Flag 1	Flag 2	Field length decimal	Default value	Type	
						Char	**32–36** The date the user ID association was approved or refused. (*yyyymmdd*)
						Int	**37–44** The time the user ID association was approved or refused. For the format of the time, see the TIME macro as documented in *z/OS MVS Programming: Assembler Services Reference, Volume 2 (IARR2V-XCTLX)* .
							45–56 Reserved for IBM's use.
						Char	**57–64** The user ID that created the entry.
CERTCT	070	10	00	00000004	00		Number of certificate names.
CERTNAME	071	80	00	00000000	00		Name of certificate. Names correspond to profiles in the DIGTCERT class for the user.
CERTLABL	072	80	00	00000000	00		Label associated with the certificate.
CERTSJDN	073	80	00	00000000	00		Subject's distinguished name.
CERTPUBK	074	80	00	00000000	00		Public key associated with the certificate.
CERTRSV3	075	80	00	00000000	00		Reserved for IBM's use.
FLAG9	076	20	80	00000001	00		Restricted Access = BIT0.
NMAPCT	077	10	00	00000004	00		Number of DIGTNMAP Mapping Profiles that specify this user ID.
NMAPLABL	078	80	00	00000000	00		Label associated with this mapping.
NMAPNAME	079	80	00	00000000	00		Name of mapping profile. The names correspond to profiles in the DIGTNMAP class.
NMAPRSV1	080	80	00	00000000	00		Reserved for IBM's use.
NMAPRSV2	081	80	00	00000000	00		Reserved for IBM's use.
NMAPRSV3	082	80	00	00000000	00		Reserved for IBM's use.
NMAPRSV4	083	80	00	00000000	00		Reserved for IBM's use.
NMAPRSV5	084	80	00	00000000	00		Reserved for IBM's use.
PWDENV	085	00	08	00000000	00	Bin	Internal form of enveloped RACF password.
PASSASIS	086	20	80	00000001	00	Bin	Identifies the user as having (bit 0 is on) or not having used a mixed case password.
PHRASE	087	04	80	00000000	FF	Bin	The password phrase associated with this user.
PHRDATE	088	00	A0	00000003	FF	Bin	The date the password phrase was last changed.
PHRGEN	089	00	00	00000001	FF	Int	Current password phrase generation number.
PHRCNT	090	10	00	00000004	00	Int	Number of old password phrases.
OLDPHRNM	091	80	00	00000001	00	Int	Generation number of password phrase.
OLDPHR	092	84	00	00000008	FF	Bin	Previous password phrase, truncated to 8 bytes.
CERTSEQN	093	00	00	00000004	00	Int	Sequence number that is incremented whenever a certificate for the user is added, deleted, or altered.
PPHENV	094	00	00	00000000	00	Bin	Internal form of enveloped RACF password phrase

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
DMAPCT	095	10	00	00000004	00		Number of IDIDMAP Mapping Profiles that specify this user ID.
DMAPLABL	096	80	00	00000000	00		Label associated with this mapping.
DMAPNAME	097	80	00	00000000	00		Name of mapping profile. The names correspond to profiles in the IDIDMAP class.
DMAPRSV1	098	80	00	00000000	00		Reserved for IBM's use.
DMAPRSV2	099	80	00	00000000	00		Reserved for IBM's use.
PWDX	100	04	80	00000000	00	Bin	Password extension
OPWDXCT	101	10	00	00000004	00	Int	Password history extension: count of entries
OPWDXGEN	102	80	00	00000001	FF	Int	Password history extension: generation number
OPWDX	103	84	00	00000000	00	Char	Password history extension: password value
PHRASEX	104	04	00	00000000	00	Bin	Password phrase extension
PHRCNTX	105	10	00	00000004	00	Int	Password phrase history extension: count of entries
OLDPHRNX	106	80	00	00000001	FF	Int	Password phrase history extension: generation number
OLDPHRX	107	84	00	00000000	00	Char	Password phrase history extension: phrase value

Field name	**Field ID**	**Flag 1**	**Flag 2**	**Combination field IDs**					**Type**	
Following are the COMBINATION fields of the USER template										
DEFDATE	000	40	00	004	000	000	000	000		Combination.
CREADATE	000	40	00	004	000	000	000	000		Fields.
OWNER	000	40	00	005	000	000	000	000		
PASSDATA	000	40	00	012	013	000	000	000		
NAME	000	40	00	014	000	000	000	000		
OLDPSWDS	000	40	00	026	027	000	000	000		
LOGINFO	000	40	00	035	036	000	000	000		
FIELD	000	40	00	038	039	040	000	000		
USERDATA	000	40	00	046	047	048	000	000		
CGDEFDAT	000	40	00	052	000	000	000	000		
CGCREADT	000	40	00	052	000	000	000	000		
CGOWNER	000	40	00	053	000	000	000	000		
TUENTRY	000	40	00	068	069	000	000	000		
CERTLIST	000	40	00	071	072	000	000	000		

Field name	Field ID	Flag 1	Flag 2	Combination field IDs					Type	
CERTLST2	000	40	00	071	072	073	074	000		
CERTLST3	000	40	00	071	072	073	000	000		
CERTSIGL	000	40	00	071	073	074	000	000		
OLDPHRES	000	40	00	091	092	000	000	000		
DMAPLST1	000	40	00	096	097	000	000	000		Combination for distributed identity.
OLDPWDX	000	40	00	102	103	000	000	000		Alias for extended password history entry.
OLDPHREX	000	40	00	106	107	000	000	000		Alias for extended password phrase history entry.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
Following is the DFP segment of the USER template							
DFP	001	00	00	00000000	00		Start of segment fields
DATAAPPL	002	00	00	00000000	00	Char	Data Application; maximum length=8
DATACLAS	003	00	00	00000000	00	Char	Data Class; maximum length=8
MGMTCLAS	004	00	00	00000000	00	Char	Management Class; maximum length=8
STORCLAS	005	00	00	00000000	00	Char	Storage Class; maximum length=8
Following is the TSO segment of the USER template							
TSO	001	00	00	00000000	00		Start of segment fields
TACCNT	002	00	00	00000000	00	Char	Default account numbers; maximum length=40
TCOMMAND	003	00	00	00000000	00	Char	Default command at logon; maximum length=80
TDEST	004	00	00	00000000	00	Char	Destination identifier; maximum length=8
THCLASS	005	00	00	00000000	00	Char	Default hold class; maximum length=1
TJCLASS	006	00	00	00000000	00	Char	Default job class
TLPROC	007	00	00	00000000	00	Char	Default logon procedure; maximum length=8
TLSIZE	008	00	00	00000004	00	Int	Logon size
TMCLASS	009	00	00	00000000	00	Char	Default message class; maximum length=1
TMSIZE	010	00	00	00000004	00	Int	Maximum region size
TOPTION	011	20	00	00000001	00	Bin	Default for mail notices and OIDcard
TPERFORM	012	00	00	00000004	00	Int	Performance group
TRBA	013	00	00	00000003	00	Bin	RBA of user's broadcast area
TSCLASS	014	00	00	00000000	00	Char	Default sysout class
TUDATA	015	00	00	00000002	00	Bin	2 bytes of hex user data
TUNIT	016	00	00	00000000	00	Char	Default unit name; maximum length=8
TUPT	017	00	00	00000000	00	Bin	Data from UPT control block
TSOSLABL	018	00	00	00000000	00	Char	Default logon SECLABEL; maximum length=8
TCONS	019	00	00	00000000	00	Char	Consoles support
Following is the CICS® segment of the USER template							
CICS	001	00	00	00000000	00		Start of segment fields

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
OPIDENT	002	00	00	00000003	00	Char	Operator identification; 1 to 3 bytes in length
OPCLASSN	003	10	00	00000004	00	Int	Count of operator class values
OPCLASS	004	80	00	00000001	00	Int	Operator class
OPPRTY	005	00	40	00000002	00	Int	Operator priority
XRFSOFF	006	20	00	00000001	00	Bin	XRF Re-signon option: • Bit 0 on = FORCE • Bit 0 off = NOFORCE
TIMEOUT	007	00	40	00000002	00	Bin	Terminal time-out value **Two 1-byte binary fields:** **Note:** 1. first one = hours (0–99) 2. second one = minutes (0–59) 3. special case: hours=0, minutes=60 treated the same as hours=1, minutes=0
RSLKEYN	008	10	00	00000004	00	Int	Count of resource security level (RSL) key values
RSLKEY	009	80	00	00000002	00	Int	RSL key value
TSLKEYN	010	10	00	00000004	00	Int	Count of transaction security level (TSL) key values
TSLKEY	011	80	00	00000002	00	Int	TSL key value
Following is the LANGUAGE segment of the USER template							
LANGUAGE	001	00	00	00000000	00		Start of segment fields
USERNL1	002	00	80	00000003	00	Char	User's primary language
USERNL2	003	00	80	00000003	00	Char	User's secondary language
Following is the OPERPARM segment of the USER template							
OPERPARM	001	00	00	00000000	00		Start of segment fields
OPERSTOR	002	00	00	00000002	00	Bin	STORAGE keyword
OPERAUTH	003	00	00	00000002	00	Bin	AUTH keyword: • X'8000' = MASTER • X'4000' = ALL • X'2000' = SYS • X'10000' = IO • X'0800' = CONS • X'0400' = INFO
OPERMFRM	004	00	00	00000002	00	Bin	MFORM keyword: • Bit 0 indicates T • Bit 1 indicates S • Bit 2 indicates J • Bit 3 indicates M • Bit 4 indicates X

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
OPERLEVL	005	00	00	00000002	00	Bin	LEVEL keyword: • Bit 0 indicates R • Bit 1 indicates I • Bit 2 indicates CE • Bit 3 indicates E • Bit 4 indicates IN • Bit 5 indicates NB • Bit 6 indicates ALL Bit 6 is mutually exclusive with all other bits except Bit 5.
OPERMON	006	00	00	00000002	00	Bin	MONITOR keyword: • Bit 0 indicates JOBNAMES • Bit 1 indicates JOBNAMEST • Bit 2 indicates SESS • Bit 3 indicates SESST • Bit 4 indicates STATUS Bits 0 and 1 are mutually-exclusive, as are bits 2 and 3.
OPERROUT	007	00	00	00000000	00	Bin	ROUTCODE keyword; 16-bit length bitstring in which each bit indicates a particular ROUTCODE.
OPERLOGC	008	00	00	00000001	00	Bin	LOGCMDRESP keyword. **Value** **Meaning when set** **X'80'** Indicates SYSTEM was specified. **X'40'** Indicates NO was specified.
OPERMGID	009	00	00	00000001	00	Bin	MIGID keyword. **Value** **Meaning when set** **X'80'** Indicates YES was specified. **X'40'** Indicates NO was specified.
OPERDOM	010	00	00	00000001	00	Bin	DOM keyword. **Value** **Meaning when set** **X'80'** Indicates NORMAL was specified. **X'40'** Indicates ALL was specified. **X'20'** Indicates NONE was specified.
OPERKEY	011	00	00	00000000	00	Bin	KEY keyword; maximum length=8
OPERCMDS	012	00	00	00000000	00	Bin	CMDSYS keyword; maximum length=8 (or '*')

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
OPERUD	013	00	00	00000001	00	Bin	UD keyword. **Value** **Meaning when set** **X'80'** Indicates YES was specified. **X'40'** Indicates NO was specified.
OPERMCNT	014	10	00	00000004	00	Bin	Count of MSCOPE systems
OPERMSCP	015	80	00	00000008	00	Bin	MSCOPE systems
OPERALTG	016	00	00	00000000	00	Bin	ALTGRP keyword **Value** **Meaning when set** **X'80'** Indicates YES was specified. **X'40'** Indicates NO was specified.
OPERAUTO	017	00	00	00000001	00	Bin	AUTO keyword; X'80' indicates YES; X'40' indicates NO.
OPERHC	018	00	00	00000001	00	BIN	HC Keyword; X'80' indicates YES; X'40' indicates NO.
OPERINT	019	00	00	00000001	00	BIN	INTIDS Keyword; X'80' indicates YES; X'40' indicates NO.
OPERUNKN	020	00	00	00000001	00	BIN	UNKNIDS Keyword; X'80' indicates YES; X'40' indicates NO.
Following is the WORK ATTRIBUTES segment of the USER template							
WORKATTR	001	00	80	00000000	00		Start of segment fields
WANAME	002	00	80	00000000	00	Char	User name for SYSOUT; maximum length=60
WABLDG	003	00	80	00000000	00	Char	Building for delivery; maximum length=60
WADEPT	004	00	80	00000000	00	Char	Department for delivery; maximum length=60
WAROOM	005	00	80	00000000	00	Char	Room for delivery; maximum length=60
WAADDR1	006	00	80	00000000	00	Char	SYSOUT address line 1; maximum length=60
WAADDR2	007	00	80	00000000	00	Char	SYSOUT address line 2; maximum length=60
WAADDR3	008	00	80	00000000	00	Char	SYSOUT address line 3; maximum length=60
WAADDR4	009	00	80	00000000	00	Char	SYSOUT address line 4; maximum length=60
WAACCNT	010	00	80	00000000	00	Char	Account number; maximum length=255
Following is the OMVS segment of the USER template							
OMVS	001	00	00	00000000	00		Start of segment fields
UID	002	00	10	00000004	FF	Int	UID
HOME	004	00	00	00000000	00	Char	HOME Path; maximum length=1023
PROGRAM	005	00	00	00000000	00	Char	Initial Program; maximum length=1023
CPUTIME	006	00	00	00000004	FF	Int	CPUTIMEMAX
ASSIZE	007	00	00	00000004	FF	Int	ASSIZEMAX
FILEPROC	008	00	00	00000004	FF	Int	FILEPROCMAX
PROCUSER	009	00	00	00000004	FF	Int	PROCUSERMAX
THREADS	010	00	00	00000004	FF	Int	THREADSMAX

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
MMAPAREA	011	00	00	00000004	FF	Int	MMAPAREAMAX
MEMLIMIT	012	00	00	00000000	0	Char	MEMLIMIT; maximum length = 9
SHMEMMAX	013	00	00	00000000	0	Char	SHMEMMAX; maximum length = 9
Following is the NETVIEW segment of the USER template							
NETVIEW	001	00	00	00000000	00		Start of segment fields
IC	002	00	00	00000000	00	Char	The command or command list to be processed by NetView® for this operator when the operator logs on to Netview; maximum length=255.
CONSNAME	003	00	00	00000000	00	Char	The default MCS console identifier; maximum length=8.
CTL	004	20	00	00000001	00	Bin	CTL keyword – Specifies whether a security check is performed for this NetView operator when they try to use a span or try to do a cross-domain logon. **Value** **Meaning when set** **X'00'** Indicates CTL was not specified or CTL(SPECIFIC) was specified. **X'80'** Indicates CTL(GLOBAL) was specified. **X'40'** Indicates CTL(GENERAL) was specified.
MSGRECVR	005	20	00	00000001	00	Bin	MSGRECVR keyword **Value** **Meaning when set** **X'00'** Indicates the operator can receive unsolicited messages that are not routed to a specific NetView operator. **X'80'** Indicates the operator cannot receive unsolicited messages that are not routed to a specific NetView operator.
OPCLASSN	006	10	00	00000004	00	Int	Count of operator class values.
OPCLASS	007	80	40	00000002	00	Int	Specifies a NetView scope class for which the operator has authority. This is a 2-byte repeating field. Each member can have fixed-binary values from 1 to 2040.
DOMAINSN	008	10	00	00000004	00	Int	The number of domains the NetView operator controls.
DOMAINS	009	80	00	00000000	00	Char	Specifies the identifier of NetView programs in another NetView domain for which this operator has authority. This is a variable length (5-character maximum) repeating field.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
NGMFADMN	010	20	00	00000001	00	Bin	NGMFADMN keyword **Value** **Meaning when set** **X'00'** The NetView operator does not have administrator authority to the NetView Graphic Monitor Facility (NGMF). **X'80'** The NetView operator has administrator authority to the NetView graphic monitor facility (NGMF).
NGMFVSPN	011	00	00	00000000	00		NetView Graphic Monitor Facility view span options; maximum length=8
Following is the DCE segment of the USER template							
DCE	001	00	00	00000000	00		Start of segment fields
UUID	002	00	00	00000036	FF	Char	User's DCE principal's UUID; exactly 36 characters, in the format *nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn* where *n* is any hexadecimal digit.
DCENAME	003	00	00	00000000	00	Char	User's DCE principal name; maximum length=1023
HOMECELL	004	00	00	00000000	00	Char	Home cell for this DCE user; maximum length=1023, and it must start with either */.../* or */.:/*
HOMEUUID	005	00	00	00000036	FF	Char	Home cell UUID; exactly 36 characters, in the format *nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn* where *n* is any hexadecimal digit.
DCEFLAGS	006	20	00	00000001	00	Bin	User flags
DPASSWDS	007	00	00	00000000	00	Char	Current DCE password
DCEENCRY	008	00	00	00000071	00	Bin	PW mask/encrypt key
Following is the OVM segment of the USER template							
OVM	001	00	00	00000000	00		Start of segment fields
UID	002	00	00	00000004	FF	Int	OVM - UID
HOME	003	00	00	00000000	00	Char	Home path; maximum length=1023
PROGRAM	004	00	00	00000000	00	Char	Initial program; maximum length=1023
FSROOT	005	00	00	00000000	00	Char	File system root; maximum length=1023
Following is the LNOTES segment of the USER template							
LNOTES	001	00	00	00000000	00		Start of segment fields
SNAME	002	00	14	00000000	00	Char	User's short name; maximum length=64
Following is the NDS segment of the USER template							
NDS	001	00	00	00000000	00		Start of segment fields
UNAME	002	00	14	00000000	00	Char	User's user name; maximum length=246
Following is the KERB segment of the USER template							
KERB	001	00	00	00000000	00		Start of segment fields
KERBNAME	002	00	00	00000000	00	Char	Kerberos principal name
MINTKTLF	003	00	00	00000000	00	Char	Reserved for IBM's use.
MAXTKTLF	004	00	00	00000000	00	Char	Maximum ticket life

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
DEFTKTLF	005	00	00	00000000	00	Char	Reserved for IBM's use.
SALT	006	00	00	00000000	00	Char	Current key salt
ENCTYPE	007	00	00	00000000	00	Char	Encryption type
CURKEYV	008	00	00	00000000	00	Char	Current key version
CURKEY	009	00	00	00000000	00	Char	Current DES key
PREVKEYV	010	00	00	00000000	00	Char	Previous key version
PREVKEY	011	00	00	00000000	00	Char	Previous DES key
ENCRYPT	012	00	00	00000004	55	Bin	Encryption types for SETROPTS KERBLVL(1)
Following is the PROXY segment of the USER template							
PROXY	001	00	00	00000000	00		Start of segment fields
LDAPHOST	002	00	00	00000000	00	Char	LDAP server URL; maximum length: 1023
BINDDN	003	00	00	00000000	00	Char	Bind distinguished name; maximum length: 1023
BINDPW	004	00	08	00000000	00	Char	Bind password; maximum length: 128
BINDPWKY	005	00	08	00000071	00	Char	Bind password mask or encrypt key
Following is the EIM segment of the USER template							
EIM	001	00	00	00000000	00	Char	Start of segment fields
LDAPPROF	1	00	00	00000000	00	Char	LDAPBIND profile name

Connect Template for the Restructured Database

The connect template is included to maintain compatibility with previous releases. You can continue to code macros to manipulate CONNECT data. This template is provided to show what fields continue to be supported. Information that was formerly stored in CONNECT profiles was moved to the USER profile. The information is in the CGGRPCT repeat group, and the fields are prefixed by “CG”.

NOT programming interface information

REVOKEDT
RESUMEDT

End of NOT programming interface information

Note: Application developers should not depend on being able to use RACROUTE REQUEST=EXTRACT for the BASE segment fields on any security product other than RACF. These products are expected to support only such segments as DFP and TSO.

The contents of the connect template are as follows:

Template							**Field being described**
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
CONNECT	001	00	00	00000000			
ENTYPE	002	00	00	00000001		Int	The number (3) corresponding to connect profiles.
VERSION	003	00	00	00000001		Int	The version field from the profile. Always X'01'.
AUTHDATE	004	00	A0	00000003		Date	The date the user was connected to the group.
AUTHOR	005	00	80	00000008		Char	The owner (user ID) of the connect entry.
LJTIME	006	01	00	00000004		Time	The time that RACROUTE REQUEST=VERIFY was last issued for this user and group.
LJDATE	007	01	20	00000003		Date	The date that RACROUTE REQUEST=VERIFY was last issued for this user and group.
UACC	008	20	80	00000001		Bin	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** — EXECUTE access **5–6** — Reserved for IBM's use **7** — EXECUTE access
INITCNT	009	01	00	00000002		Int	The number of RACROUTE REQUEST=VERIFY macro instructions issued for this user and group.
FLAG1	010	20	80	00000001		Bin	Identifies the user as having (bit 0 is on) or not having the ADSP attribute.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
FLAG2	011	20	80	00000001		Bin	Identifies the user as having (bit 0 is on) or not having the group-SPECIAL attribute.
FLAG3	012	20	80	00000001		Bin	Identifies the user as having (bit 0 is on) or not having the group-OPERATIONS attribute.
FLAG4	013	20	80	00000001		Bin	Identifies the user as having (bit 0 is on) or not having the REVOKE attribute.
FLAG5	014	20	80	00000001		Bin	Identifies the user as having (bit 0 is on) or not having the GRPACC attribute.
NOTRMUAC	015	20	80	00000001		Bin	Identifies whether 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.
GRPAUDIT	016	20	80	00000001		Bin	Identifies the user as having (bit 0 is on) or not having the group-AUDITOR attribute.
REVOKEDT	017	00	20	00000000		Date	The date the user will be revoked. This field either has length 0, or contains a 3-byte revoke date.
RESUMEDT	018	00	20	00000000		Date	The date the user will be resumed. This field either has length 0, or contains a 3-byte resume date.

Field name	**Field ID**	**Flag 1**	**Flag 2**	**Combination field IDs**					**Type**	
The following are the COMBINATION fields.										
DEFDATE	000	40	00	004	000	000	000	000	Char	Combination.
CREADATE	000	40	00	004	000	000	000	000	Char	Fields.
OWNER	000	40	00	005	000	000	000	000	Char	

Data Set Template for the Restructured Database

The data set template describes the fields of the data set profiles in a RACF database.

NOT programming interface information

ACL2VAR
AUDITQF
AUDITQS
CATEGORY
FIELD
FLDCNT
FLDFLAG
FLDNAME
FLDVALUE
NUMCTGY

End of NOT programming interface information

Note: Application developers should not depend on being able to use RACROUTE REQUEST=EXTRACT for the BASE segment fields on any security product other than RACF. These products are expected to support only such segments as DFP and TSO.

The contents of the data set template are as follows:

Template							**Field being described**
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
DATASET	001	00	00	00000000	00		
ENTYPE	002	00	00	00000001	04	Int	The number (4) corresponding to data set profiles.
VERSION	003	00	00	00000001	01	Int	The version field from the profile. Always X'01'.
CREADATE	004	00	20	00000003	FF	Date	The date the data set was initially defined to RACF; 3-byte date.
AUTHOR	005	00	00	00000008	FF	Char	The owner of the data set.
LREFDAT	006	01	20	00000003	FF	Date	The date the data set was last referenced; 3-byte date.
LCHGDAT	007	01	20	00000003	FF	Date	The date the data set was last updated; 3-byte date.
ACSALTR	008	01	00	00000002	FF	Int	The number of times the data set was accessed with ALTER authority.
ACSCNTL	009	01	00	00000002	FF	Int	The number of times the data set was accessed with CONTROL authority.
ACSUPDT	010	01	00	00000002	FF	Int	The number of times the data set was accessed with UPDATE authority.
ACSREAD	011	01	00	00000002	FF	Int	The number of times the data set was accessed with READ authority.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
UNIVACS	012	20	00	00000001	00	Bin	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** EXECUTE access **5–6** Reserved for IBM's use **7** EXECUTE access
FLAG1	013	20	00	00000001	00	Bin	Identifies whether the data set is a group data set. If bit 0 is on, the data set is a group data set.
AUDIT	014	20	00	00000001	00	Bin	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 for IBM's use
GROUPNM	015	00	00	00000008	FF	Char	The current connect group of the user who created this data set.
DSTYPE	016	20	00	00000001	00	Bin	Identifies the data set as a VSAM, non-VSAM (or generic), 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 for IBM's use
LEVEL	017	00	00	00000001	FF	Int	Data set level.
DEVTYP	018	00	00	00000004	FF	Bin	The type of device on which the data set resides; only for non-model, discrete data sets.
DEVTYPX	019	00	00	00000008	FF	Char	The EBCDIC name of the device type on which the data set resides; only for non-model, discrete data sets.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
GAUDIT	020	20	00	00000001	00	Bin	Global audit flags. (Audit options specified by a user with the AUDITOR or group-AUDITOR attribute.) **Bit** **Meaning when set** **0** Audit all accesses **1** Audit successful accesses **2** Audit accesses that fail **3** No auditing **4–7** Reserved for IBM's use
INSTDATA	021	00	00	00000000	00	Char	Installation data; maximum length=255.
GAUDITQF	025	00	00	00000001	FF	Bin	Global audit FAILURES qualifier. The AUDITQS, AUDITQF, GAUDITQS, and GAUDITQF fields have the following format: **Value** **Meaning when set** **X'00'** Log access at READ level **X'01'** Log access at UPDATE level **X'02'** Log access at CONTROL level **X'03'** Log access at ALTER level
AUDITQS	022	00	00	00000001	FF	Bin	Audit SUCCESS qualifier.
AUDITQF	023	00	00	00000001	FF	Bin	Audit FAILURES qualifier.
GAUDITQS	024	00	00	00000001	FF	Bin	Global audit SUCCESS qualifier.
WARNING	026	20	00	00000001	00	Bin	Identifies the data set as having (bit 7 is on) or not having the WARNING attribute.
SECLEVEL	027	00	00	00000001	FF	Int	Data set security level.
NUMCTGY	028	10	00	00000004	00	Int	The number of categories.
CATEGORY	029	80	00	00000002	00	Bin	A list of numbers corresponding to the categories to which this data set belongs.
NOTIFY	030	00	00	00000000	00	Char	User to be notified when access violations occur against a data set protected by this profile.
RETPD	031	00	00	00000000	00	Int	The number of days protection is provided for the data set. If used, the field will be a two-byte binary number.
ACL2CNT	032	10	00	00000004	00	Int	The number of program and user combinations currently authorized to access the data set.
PROGRAM	033	80	00	00000008	00	Char	The name of a program currently authorized to access the data set, or a 1-byte flag followed by 7 bytes reserved for IBM's use.
USER2ACS	034	80	00	00000008	00	Char	User ID or group.
PROGACS	035	80	00	00000001	00	Bin	The access authority of the program and user combinations.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
PACSCNT	036	80	00	00000002	00	Int	Access count.
ACL2VAR	037	80	00	00000000	00	Char	Additional conditional data, 9-byte length, in which the the 1st byte tells what kind of access is allowed and the remaining 8 bytes contain the data.
FLDCNT	038	10	00	00000004	00		Reserved for IBM's use.
FLDNAME	039	80	00	00000008	00		Reserved for IBM's use.
FLDVALUE	040	80	00	00000000	00		Reserved for IBM's use.
FLDFLAG	041	A0	00	00000001	00		Reserved for IBM's use.
VOLCNT	042	10	00	00000004	00	Int	The number of volumes containing the data set.
VOLSER	043	80	00	00000006	00	Char	A list of the serial numbers of the volumes containing the data set.
ACLCNT	044	10	00	00000004	00	Int	The number of users and groups currently authorized to access the data set.
USERID	045	80	00	00000008	00	Char	The user ID or group name of each user or group authorized to access the data set.
USERACS	046	A0	00	00000001	00	Bin	The access authority that each user or group has for the data set.
							Bit **Meaning when set** **0** ALTER access **1** CONTROL access **2** UPDATE access **3** READ access **4** EXECUTE access **5–6** Reserved for IBM's use **7** NONE access
ACSCNT	047	80	00	00000002	00	Int	The number of times the data set was accessed by each user or group.
USRCNT	048	10	00	00000004	00	Int	Reserved for installation's use.
USRNM	049	80	00	00000008	00		Reserved for installation's use.
USRDATA	050	80	00	00000000	00		Reserved for installation's use.
USRFLG	051	A0	00	00000001	00		Reserved for installation's use.
SECLABEL	052	00	00	00000008	00	Char	Security label.

Field name	**Field ID**	**Flag 1**	**Flag 2**	**Combination field IDs**					**Type**	
Following are the COMBINATION fields of the Data Set Template										
DEFDATE	000	40	00	004	000	000	000	000	Char	Combination.
AUTHDATE	000	40	00	004	000	000	000	000	Char	Fields.
OWNER	000	40	00	005	000	000	000	000	Char	
UACC	000	40	00	012	000	000	000	000		

Field name	Field ID	Flag 1	Flag 2	Combination field IDs					Type
ACL2	000	40	00	033	034	035	036	037	
ACL2A3	000	40	00	033	034	035	037	000	
ACL2A2	000	40	00	033	034	035	036	000	
ACL2A1	000	40	00	033	034	035	000	000	
FIELD	000	40	00	039	040	041	000	000	
VOLUME	000	40	00	043	000	000	000	000	
ACL	000	40	00	045	046	047	000	000	
ACL1	000	40	00	045	046	000	000	000	
USERDATA	000	40	00	049	050	051	000	000	

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
Following is the DFP segment of the Data Set Template							
DFP	001	00	00	00000000	00		Start of segment fields
RESOWNER	002	00	00	00000008	FF	Char	Resource owner; must represent a user ID or group name

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
Following is the TME segment of the Data Set Template							
TME	001	00	00	00000000	00		Start of segment fields
ROLEN	002	10	00	00000004	00	Int	Count of role-access specifications
ROLES	003	80	00	00000000	00	Char	Role-access specifications

General Template for the Restructured Database

The general template describes the fields of general resource profiles in a RACF database.

NOT programming interface information

ACL2RSVD AUDITQF
AUDITQS CATEGORY
CURKEY CURKEYV
ENCTYPE
FIELD
FLDCNT
FLDFLAG
FLDNAME
FLDVALUE
GAUDITQF
GAUDITQS
MEMCNT
MEMLIST
NUMCTGY
PREVKEY
PREVKEYV
RACLDSP
RACLHDR
SALT
SSKEY

End of NOT programming interface information

Note: Application developers should not depend on being able to use RACROUTE REQUEST=EXTRACT for the BASE segment fields on any security product other than RACF. These products are expected to support only such segments as DFP and TSO.

The contents of the general template are as follows:

Template							**Field being described**
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
The following is the BASE segment of the GENERAL template.							
GENERAL	001	00	00	00000000	00		
ENTYPE	002	00	00	00000001	05	Int	The number (5) corresponding to profiles for resources defined in the class descriptor table.
VERSION	003	00	00	00000001	01	Int	The version field from the profile. Always X'01'.
CLASTYPE	004	00	00	00000001	FF	Int	The class to which the resource belongs (from the ID=class-number operand of the ICHERCDE macro).
DEFDATE	005	00	20	00000003	FF	Date	The date the resource was defined to RACF.
OWNER	006	00	00	00000008	FF	Char	The owner of the resource.
LREFDAT	007	01	20	00000003	FF	Date	The date the resource was last referenced.
LCHGDAT	008	01	20	00000003	FF	Date	The date the resource was last updated.
ACSALTR	009	01	00	00000002	FF	Int	The number of times the resource was accessed with ALTER authority.
ACSCNTL	010	01	00	00000002	FF	Int	The number of times the resource was accessed with CONTROL authority.
ACSUPDT	011	01	00	00000002	FF	Int	The number of times the resource was accessed with UPDATE authority.
ACSREAD	012	01	00	00000002	FF	Int	The number of times the resource was accessed with READ authority.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
UACC	013	20	80	00000001	00	Bin	The universal access authority for the resource. **Bit** **Meaning when set** **0** ALTER access **1** CONTROL access **2** UPDATE access **3** READ access **4** EXECUTE access **5–6** Reserved for IBM's use **7** NONE access.
AUDIT	014	20	00	00000001	00	Bin	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 for IBM's use
LEVEL	015	20	00	00000001	00	Int	Resource level.
GAUDIT	016	20	00	00000001	00	Bin	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 for IBM's use
INSTDATA	017	00	00	00000000	00	Char	Installation data; maximum length=255.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
GAUDITQF	021	00	00	00000001	FF	Bin	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
AUDITQS	018	00	00	00000001	FF	Bin	Audit SUCCESS qualifier. (Audit options specified by a user with the AUDITOR or group-AUDITOR attribute.) **Bit** **Meaning when set** **0** Audit all accesses **1** Audit successful accesses **2** Audit accesses that fail **3** No auditing **4–7** Reserved for IBM's use
AUDITQF	019	00	00	00000001	FF	Bin	Audit FAILURES qualifier. (Audit options specified by a user with the AUDITOR or group-AUDITOR attribute.) **Bit** **Meaning when set** **0** Audit all accesses **1** Audit successful accesses **2** Audit accesses that fail **3** No auditing **4–7** Reserved for IBM's use

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
GAUDITQS	020	00	00	00000001	FF	Bin	Global audit SUCCESS qualifier. (Audit options specified by a user with the AUDITOR or group-AUDITOR attribute.) **Bit** **Meaning when set** **0** Audit all accesses **1** Audit successful accesses **2** Audit accesses that fail **3** No auditing **4–7** Reserved for IBM's use
WARNING	022	20	00	00000001	00	Bin	Identifies the data set as having (bit 7 is on) or not having the WARNING attribute.
RESFLG	023	20	00	00000001	00	Bin	Resource profile flags: **Bit** **Meaning when set** **0** TAPEVOL can only contain one data set. **1** TAPEVOL profile is automatic. **2** Maintain TVTOC for TAPEVOL. **3–7** Reserved for IBM's use
TVTOCCNT	024	10	00	00000004	00	Int	The number of TVTOC entries.
TVTOCSEQ	025	80	00	00000002	00	Int	The file sequence number of tape data set.
TVTOCCRD	026	80	20	00000003	00	Date	The date the data set was created.
TVTOCIND	027	A0	00	00000001	00	Bin	Data set profiles flag (RACF indicator bit): **Bit** **Meaning when set** **1** Discrete data set profile exists **2–7** Reserved for IBM's use
TVTOCDSN	028	80	00	00000000	00	Char	The RACF internal name.
TVTOCVOL	029	80	00	00000000	00	Char	This field is a list of the volumes on which the tape data set resides.
TVTOCRDS	030	80	00	00000000	00	Char	The name used when creating the tape data set; maximum length=255.
NOTIFY	031	00	00	00000000	00	Char	The user to be notified when access violations occur against resource protected by this profile.
LOGDAYS	032	20	00	00000001	00	Bin	The days of the week the TERMINAL cannot be used. (Bit 0 equals Sunday, bit 1 equals Monday, and so on).
LOGTIME	033	00	00	00000000	00	Time	The time of the day the TERMINAL can be used.
LOGZONE	034	00	00	00000000	00	Bin	The time zone in which the terminal is located.
NUMCTGY	035	10	00	00000004	00	Int	Number of categories.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
CATEGORY	036	80	00	00000002	00	Int	List of categories.
SECLEVEL	037	00	00	00000001	FF	Int	Resource security level.
FLDCNT	038	10	00	00000004	00	Int	Reserved for IBM's use.
FLDNAME	039	80	00	00000008	00		Reserved for IBM's use.
FLDVALUE	040	80	00	00000000	00		Reserved for IBM's use.
FLDFLAG	041	A0	00	00000001	00		Reserved for IBM's use.
APPLDATA	042	00	00	00000000	00	Char	Application data.
MEMCNT	043	10	80	00000004	00	Int	The number of members.
MEMLST	044	80	80	00000000	00	Bin	The resource group member. For SECLABEL class, a 4-byte SMF ID.
VOLCNT	045	10	00	00000004	00	Int	Number of volumes in tape volume set.
VOLSER	046	80	00	00000006	00	Char	Volume serials of volumes in tape volume set.
ACLCNT	047	10	80	00000004	00	Int	The number of users and groups currently authorized to access the resource.
USERID	048	80	80	00000008	00	Char	The user ID or group name of each user or group authorized to access the resource.
USERACS	049	A0	80	00000001	00	Bin	The access authority that each user or group has for the resource. **Bit** **Meaning when set** **0** ALTER access **1** CONTROL access **2** UPDATE access **3** READ access **4** EXECUTE access **5–6** Reserved for IBM's use **7** NONE access **Note:** Each of the above access authority fields have mutually-exclusive bits with the exception of EXECUTE+NONE.
ACSCNT	050	80	00	00000002	00	Int	The number of times the resource was accessed by each user or group.
USRCNT	051	10	00	00000004	00	Int	Reserved for installation's use.
USRNM	052	80	00	00000008	00		Reserved for installation's use.
USRDATA	053	80	00	00000000	00		Reserved for installation's use.
USRFLG	054	A0	00	00000001	00		Reserved for installation's use.
SECLABEL	055	00	00	00000008	00	Char	Security label.
ACL2CNT	056	10	00	00000004	00	Int	Number of entries in conditional access list.
ACL2NAME	057	80	00	00000008	00	Bin	1 indicator byte; 7 bytes reserved for IBM's use.
ACL2UID	058	80	00	00000008	00	Char	User ID or group.
ACL2ACC	059	80	00	00000001	00	Bin	Access authority.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
ACL2ACNT	060	80	00	00000002	00	Int	Access count.
ACL2RSVD	061	80	00	00000000	00	Bin	Conditional data. Reserved for IBM's use.
RACLHDR	062	00	00	00000020	00	Bin	RACGLIST header.
RACLDSP	063	00	00	00000000	00	Bin	RACGLIST dataspace information.
FILTERCT	064	10	00	00000004	00		Number of names that Hash to this DIGTNMAP Profile.
FLTRLABL	065	80	00	00000000	00		Label associated with this DIGTNMAP Mapping (matches NMAPLABL for user named by FLTRUSER or user irrmulti.)
FLTRSTAT	066	A0	00	00000001	00		Trust status – bit 0 on for trusted.
FLTRUSER	067	80	00	00000000	00		User ID or criteria profile name.
FLTRNAME	068	80	00	00000000	00		Unhashed issuer's name filter used to create this profile name, (max of 255), followed by a separator, (X'4A'), and the unhashed subject's name filter used to create this profile name, (max of 255).
FLTRSVD1	069	80	00	00000000	00		Reserved for IBM's use.
FLTRSVD2	070	80	00	00000000	00		Reserved for IBM's use.
FLTRSVD3	071	80	00	00000000	00		Reserved for IBM's use.
FLTRSVD4	072	80	00	00000000	00		Reserved for IBM's use.
FLTRSVD5	073	80	00	00000000	00		Reserved for IBM's use.
RACDHDR	074	00	08	00000000	00	Bin	CACHECLS header.

Field name	**Field ID**	**Flag 1**	**Flag 2**	**Combination field IDs**					**Type**	
Following is the COMBINATION segment of the GENERAL template.										
CREADATE	000	40	00	005	000	000	000	000		Combination.
AUTHDATE	000	40	00	005	000	000	000	000		Fields.
AUTHOR	000	40	00	006	000	000	000	000		
TVTOC	000	48	00	025	026	027	028	029		
	000	40	00	030	000	000	000	000		
LOGINFO	000	40	00	032	033	034	000	000		
FIELD	000	40	00	039	040	041	000	000		
ACL	000	40	00	048	049	050	000	000		
ACL1	000	40	00	048	049	000	000	000		
USERDATA	000	40	00	052	053	054	000	000		
ACL2	000	40	00	057	058	059	060	061		Conditional access list
ACL2A3	000	40	00	057	058	059	060	000		Conditional access list
FLTRLST1	000	40	00	065	066	067	068	000		Combo field for FILTER
FLTRLST2	000	40	00	065	067	068	000	000		Combo field for FILTER
CERTRING	000	40	00	010	011	009	000	000		Digital Certificate Data.
CERTRNG2	000	40	00	009	011	000	000	000		
CERTRNG3	000	40	00	009	012	013	000	000		

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
Following is the SESSION segment of the GENERAL template.							
SESSION	001	00	00	00000000	00		Start of segment fields
SESSKEY	002	00	00	00000000	00	Bin	Session key; maximum length = 8
SLSFLAGS	003	20	00	00000001	00	Bin	Session flag byte **Bit** **Meaning when set** **0** SLSLOCK—This profile is locked out **1–7** Reserved for IBM's use
KEYDATE	004	00	00	00000004	00	Date	Last date session key was changed. It is in the format *0cyyddF* where *c*=0 for 1900–1999 and *c*=1 for 2000–2099. For more information on this MVS-returned format, see *z/OS MVS Programming: Assembler Services Guide* .
KEYINTVL	005	00	00	00000002	00	Int	Number of days before session key expires
SLSFAIL	006	00	00	00000002	00	Int	Current number of invalid attempts
MAXFAIL	007	00	00	00000002	00	Int	Number of invalid attempts before lockout
SENTCNT	008	10	00	00000004	00	Int	Number of session entities in list
SENTITY	009	80	00	00000035	00	Char	Entity name
SENTFLCT	010	80	00	00000002	00	Int	Number of failed attempts for this entity
CONVSEC	011	20	00	00000001	00	Bin	Conversation security. **Value** **Meaning** **X'40'** Conversation security **X'50'** Persistent verification **X'60'** User ID and password already verified **X'70'** User ID and password already verified plus persistent verification **X'80'** Security none
Following is the DLFDATA segment of the GENERAL template.							
DLFDATA	001	00	00	00000000	00		Start of segment fields
RETAIN	002	20	00	00000001	00	Bin	Retain flag byte
JOBNMCNT	003	10	00	00000004	00	Int	Count of jobnames
JOBNAMES	004	80	00	00000000	00	Char	Jobnames; maximum length=8
Following is the SSIGNON segment of the GENERAL template.							
SSIGNON	001	00	00	00000000	00		Start of segment fields
SSKEY	002	00	00	00000000	00	Bin	Secured signon key
Following is the STDATA segment of the GENERAL template.							
STDATA	001	00	00	00000000	00		Start of segment fields
STUSER	002	00	00	00000008	40	Char	User ID or =MEMBER
STGROUP	003	00	00	00000008	40	Char	Group name or =MEMBER

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
FLAGTRUS	004	20	00	00000001	00	Bin	Trusted flag, X'80' = trusted
FLAGPRIV	005	20	00	00000001	00	Bin	Privileged flag, X'80' = privileged
FLAGTRAC	006	20	00	00000001	00	Bin	Trace usage flag X'80' = issue IRR8I2I
Following is the SVFMR segment of the GENERAL template.							
SVFMR	001	00	00	00000000	00		Start of segment fields
SCRIPTN	002	00	00	00000008	00	Char	Script name
PARMN	003	00	00	00000008	00	Char	Parameter name
Following is the CERTDATA segment of the GENERAL template.							
CERTDATA	001	00	00	00000000	00		Start of segment fields
CERT	002	00	00	00000000	00	Bin	Digital Certificate
CERTPRVK	003	00	00	00000000	00	Bin	Private key or key label
RINGCT	004	10	00	00000004	00	Int	Number of key rings associated with this certificate
RINGNAME	005	80	00	00000000	00	Char	Profile name of a ring with which this certificate is associated
CERTSTRT	006	00	00	00000000	00		Date and time from which the certificate is valid. This is an 8-byte TOD format field.
CERTEND	007	00	00	00000000	00		Date and time after which the certificate is not valid. This is an 8-byte TOD format field.
							The CERTCT repeat group identifies the certificates that are associated with a key ring. It is used **only** with DIGTRING profiles.
CERTCT	008	10	00	00000004	00	Int	The number of certificates associated with this key ring
CERTNAME	009	80	00	00000000	00	Char	The profile name of the certificate
CERTUSAG	010	80	00	00000004	00	Bin	Certificate usage in ring: • X'00000000' – PERSONAL • X'00000001' – SITE • X'00000002' – CERTAUTH
CERTDFLT	011	80	00	00000001	00	Bin	Verifies if it is the default certificate: • X'00' – Not the default • X'80' – The default
CERTSJDN	012	80	00	00000000	00	Bin	The subject name of the entity to whom the certificate is issued. This field is a BER-encoded format of the subject's distinguished name as contained in the certificate
CERTLABL	013	80	00	00000000	00	Char	Label associated with the certificate
CERTRSV1	014	80	00	00000000	00		Reserved for IBM's use.
CERTRSV2	015	80	00	00000000	00		Reserved for IBM's use.
CERTRSV3	016	80	00	00000000	00		Reserved for IBM's use.
CERTRSV4	017	80	00	00000000	00		Reserved for IBM's use.
CERTRSV5	018	80	00	00000000	00		Reserved for IBM's use.
CERTRSV6	019	80	00	00000000	00		Reserved for IBM's use.
CERTRSV7	020	80	00	00000000	00		Reserved for IBM's use.
CERTRSV8	021	80	00	00000000	00		Reserved for IBM's use.

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
CERTRSV9	022	80	00	00000000	00		Reserved for IBM's use.
CERTRSVA	023	80	00	00000000	00		Reserved for IBM's use.
CERTRSVB	024	80	00	00000000	00		Reserved for IBM's use.
CERTRSVC	025	80	00	00000000	00		Reserved for IBM's use.
CERTRSVD	026	80	00	00000000	00		Reserved for IBM's use.
CERTRSVE	027	80	00	00000000	00		Reserved for IBM's use.
CERTRSVF	028	80	00	00000000	00		Reserved for IBM's use.
CERTRSVG	029	80	00	00000000	00		Reserved for IBM's use.
CERTRSVH	030	80	00	00000000	00		Reserved for IBM's use.
CERTRSVI	031	80	00	00000000	00		Reserved for IBM's use.
CERTRSVJ	032	80	00	00000000	00		Reserved for IBM's use.
CERTRSVK	033	80	00	00000000	00		Reserved for IBM's use.
CERTPRVT	034	00	00	00000004	00	Bin	Associated key type: • X'00000000' – No associated key, • X'00000001' – PKCS DER-encoded, • X'00000002'> – ICSF token label, • X'00000003' – PCICC label, • X'00000004' – DSA, • >X'00000005' – ICSF public token label
CERTPRVS	035	00	00	00000004	00	Int	Private key size in bits
CERTLSER	036	00	00	00000008	00	Bin	The low order 8 bytes of the last certificate that was signed with this key. This field is used with DIGTCERT profiles only
RINGSEQN	037	00	00	00000004	00	Int	Ring change count
Following is the TME segment of the GENERAL template.							
TME	001	00	00	00000000	00		Start of segment fields
PARENT	002	00	00	00000000	00	Char	Parent name
CHILDN	003	10	00	00000004	00	Int	Count of children
CHILDREN	004	80	00	00000000	00	Char	Child names
RESN	005	10	00	00000004	00	Int	Count of resource-access specifications
RESOURCE	006	80	00	00000000	00		Resource-access specifications
GROUPN	007	10	00	00000004	00	Int	Count of groups
GROUPS	008	80	00	00000008	00		Group names
ROLEN	009	10	00	00000004	00	Int	Count of role-access specifications
ROLES	010	80	00	00000000	00	Char	Role-access specifications
Following is the KERB segment of the GENERAL template							
KERB	001	00	00	00000000	00		Start of segment fields
KERBNAME	002	00	00	00000000	00	Char	Kerberos realm name
MINTKTLF	003	00	00	00000000	00	Char	Minimum ticket life
MAXTKTLF	004	00	00	00000000	00	Char	Maximum ticket life
DEFTKTLF	005	00	00	00000000	00	Char	Default ticket life
SALT	006	00	00	00000000	00	Char	Current key salt

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
ENCTYPE	007	00	00	00000000	00	Char	Encryption type
CURKEYV	008	00	00	00000000	00	Char	Current key version
CURKEY	009	00	00	00000000	00	Char	Current DES key
PREVKEYV	010	00	00	00000000	00	Char	Previous key version
PREVKEY	011	00	00	00000000	00	Char	Previous DES key
ENCRYPT	012	00	00	00000004	55	Char	Encryption types for SETROPTS KERBLVL(1)
Following is the PROXY segment of the GENERAL template							
PROXY	001	00	00	00000000	00		Start of segment fields
LDAPHOST	002	00	00	00000000	00	Char	LDAP server URL; maximum length: 1023
BINDDN	003	00	00	00000000	00	Char	Bind distinguished name; maximum length: 1023
BINDPW	004	00	08	00000000	00	Char	Bind password; maximum length: 128
BINDPWKY	005	00	08	00000071	00	Char	Bind password mask or encrypt key
Following is the EIM segment of the GENERAL template							
EIM	001	00	00	00000000	00		Start of segment fields
DOMAINDN	002	00	00	00000000	00	Char	EIM Domain Distinguished Names
OPTIONS	003	00	00	00000004	55	Char	EIM Options
LOCALREG	004	00	00	00000000	00	Char	Local Registry Name
KERBREG	005	00	00	00000000	00	Char	Kerberos Registry Name
X509REG	006	00	00	00000000	00	Char	X509 Registry Name
Following is the ALIAS segment of the GENERAL template							
ALIAS	001	00	00	00000000	00		Start of segment fields
IPLOOK	002	00	10	00000016	00	Bin	IP lookup value
Following is the CDTINFO segment of the GENERAL template							
CDTINFO	001	00	00	0	0		Start of segment fields
CDTPOSIT	002	00	00	4	FF	Int	POSIT number for class
CDTMAXLN	003	00	00	1	8	Int	Maximum length of profile names
CDTMAXLX	004	00	00	4	FF	Int	Maximum resource or profile name length when using ENTITYX
CDTDFTRC	005	00	00	1	4	Int	Default return code
CDTKEYQL	006	00	00	4	0	Int	Number of key qualifiers
CDTGROUP	007	00	00	8	0	Char	Resource grouping class name
CDTMEMBR	008	00	00	8	0	Char	Member class name

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
CDTFIRST	009	00	00	1	X'C0'	Bin	Character restriction for first character of profile name **Value** **Meaning** **X'80'** Alphabetic **X'40'** National **X'20'** Numeric **X'10'** Special
CDTOTHER	010	00	00	1	X'C0'	Bin	Character restriction for characters of the profile name other than the first character **Value** **Meaning** **X'80'** Alphabetic **X'40'** National **X'20'** Numeric **X'10'** Special
CDTOPER	011	00	00	1	X'00'	Bin	Operations attribute considered **Value** **Meaning** **X'80'** RACF considers OPERATIONS attribute
CDTUACC	012	00	00	1	X'01'	Bin	Default UACC **Value** **Meaning** **X'80'** ALTER **X'40'** CONTROL **X'20'** UPDATE **X'10'** READ **X'08'** EXECUTE **X'04'** UACC from ACEE **X'01'** NONE

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
CDTRACL	013	00	00	1	X'00'	Bin	SETROPTS RACLIST **Value** **Meaning** **X'00'** RACLIST disallowed **X'80'** RACLIST allowed **X'40'** RACLIST required
CDGENL	014	00	00	1	X'00'	Bin	SETROPTS GENLIST **Value** **Meaning** **X'80'** GENLIST allowed
CDTPRFAL	015	00	00	1	X'80'	Bin	Profiles allowed **Value** **Meaning** **X'80'** Profiles are allowed
CDTSLREQ	016	00	00	1	X'00'	Bin	Security labels required **Value** **Meaning** **X'80'** Security labels are required
CDTMAC	017	00	00	1	X'80'	Bin	Mandatory access checking (MAC) processing **Value** **Meaning** **X'80'** Normal mandatory access checks **X'40'** Reverse mandatory access checks **X'20'** Equal mandatory access checks
CDTSIGL	018	00	00	1	X'00'	Bin	ENF Signal **Value** **Meaning** **X'80'** ENF signal to be sent
CDTCASE	019	00	00	1	X'00'	Bin	Case of profile names **Value** **Meaning** **X'00'** Upper case **X'80'** ASIS – preserve case
CDTGEN	020	00	00	1	X'80'	Bin	SETROPTS GENERIC **Value** **Meaning** **X'80'** GENERIC allowed

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
Following is the ICTX segment of the GENERAL template							
ICTX	001	00	00	00000000	00		Start of segment fields
USEMAP	002	00	00	00000001	80	Bin	Application supplied mapping **Value** **Meaning** **X'80'** Use the mapping
DOMAP	003	00	00	00000001	00	Bin	Identity cache mapping **Value** **Meaning** **X'80'** Do the mapping
MAPREQ	004	00	00	00000001	00	Bin	**Value** **Meaning** **X'80'** Mapping is required
MAPTIMEO	005	00	00	00000002	00	Int	Mapping timeout adjustment

Reserved Templates for the Restructured Database

Five unused templates are defined for future use. The installation must leave this space reserved and not use it.

The contents of the reserved template are as follows:

Template							Field being described
Field name (character data)	**Field ID**	**Flag 1**	**Flag 2**	**Field length decimal**	**Default value**	**Type**	
RSVTMP03	001	00	00	00000000	00		
ENTYPE	002	00	00	00000001	00		The number corresponding to the type of profile being described.
VERSION	003	00	00	00000001	00		Template version number.

Appendix C. RACROUTE Interface to an External Security Manager Product (Non-RACF) on z/VM

The following section contains guidance information for programmers who are implementing a non-RACF external security manager product.

If RACF is not present in the system, an installation can use an external (non-RACF) security product to provide system-security functions. An external security product on z/VM must supply the following items to provide RACROUTE functions equivalent to RACF:

- RPIUCMS module, to initialize the CMS environment in which RACROUTE requests will be issued
- RPIATGCS module, to initialize the GCS environment in which RACROUTE requests will be issued (and an RPIGCS LOADLIB, which holds this module)
- RACROUTE support code, to process the RACROUTE requests.

Details follow on the requirements for providing these functions.

Providing RPIUCMS Module

On z/VM, before issuing RACROUTE requests, a virtual machine must establish an environment in which the requests can be processed. To establish this environment in CMS, the virtual machine executes the RPIUCMS module as shown:

```
RPIUCMS INIT
```

When all RACROUTE requests are complete, the CMS virtual machine executes the RPIUCMS module as shown:

```
RPIUCMS TERM
```

Upon entry to RPIUCMS, register 1 points to a tokenized CMS parameter list. For more information, see *Application Development Guide*.

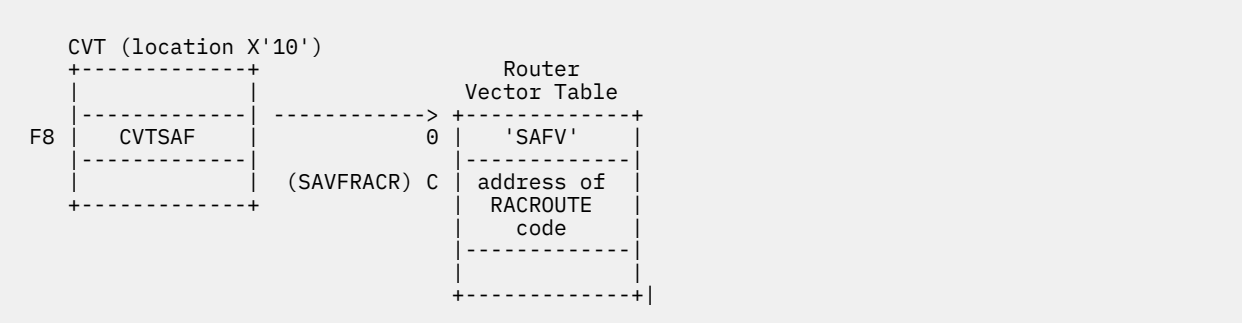
For the CMS environment the external security product must provide, at a minimum, the following functions in module RPIUCMS:

- For the RPIUCMS INIT function:
  - RPIUCMS must place the entry-point address of the RACROUTE support code in a field within the CMS nucleus of the service machine, as follows:
    1. Read the contents of the fullword at displacement 16 (X'10') in virtual storage. That fullword contains the address of a control block known as CVTSECT, or the communications vector table (CVT).
    2. Get four fullwords of storage for a router vector table. (The router vector table will contain the pointer to your RACROUTE support code.)
    3. Add 248 (X'F8') to the address of the CVT. The fullword at that address within the CVT, named CVTSAF, contains zeros. Replace those zeros with the address of the four fullwords of storage you acquired for the router vector table.
    4. Add 12 (X'0C') to the address of the router vector table. At this final address, store the entry point address of the RACROUTE support code. This entry point gets control when a RACROUTE request is made.
  - RPIUCMS should do whatever processing is necessary to initialize the remainder of your RACROUTE support code.

© Copyright IBM Corp. 1990, 2022

– If initialization of the RACROUTE support code is successful, RPIUCMS should return a zero return code.

The following diagram illustrates the above steps:
Establishing a CMS Environment with the RPIUCMS Module



When a RACROUTE request is issued and this environment has not been set up, the CVTSAF field is zero, and the RACROUTE macro ends with return code 4.

- For the RPIUCMS TERM function:
  - RPIUCMS should set the CVTSAF field to zero.
  - RPIUCMS should free the router vector table.
  - RPIUCMS should do whatever cleanup processing the external security manager needs.

Providing RPIATGCS Module and RPIGCS LOADLIB

On z/VM, before issuing RACROUTE requests, a virtual machine must establish an environment in order for the requests to be processed. Each GCS virtual machine must have access to supervisor state and authorized GCS functions. To establish a RACROUTE environment in GCS, each authorized member of the GCS group who will be issuing RACROUTE requests must do the following:

1. Enter the GCS command

   ```
   GLOBAL LOADLIB RPIGCS
   ```

2. Enter the GCS command

   ```
   LOADCMD RPIUGCS RPIATGCS
   ```

   This command identifies the load module RPIATGCS and assigns it a command name of RPIUGCS.

3. Enter the command

   ```
   RPIUGCS INIT
   ```

When all RACROUTE requests are complete, the GCS virtual machine executes the RPIUGCS command, as shown:

```
RPIUGCS TERM
```

Upon entry to RPIATGCS, register 0 and register 1 point to input parameter lists that are set up by the GCS command LOADCMD. For more information, see the description of the LOADCMD in the *z/VM: Group Control System*.

The external security product must provide, at a minimum, the following functions in module RPIATGCS:

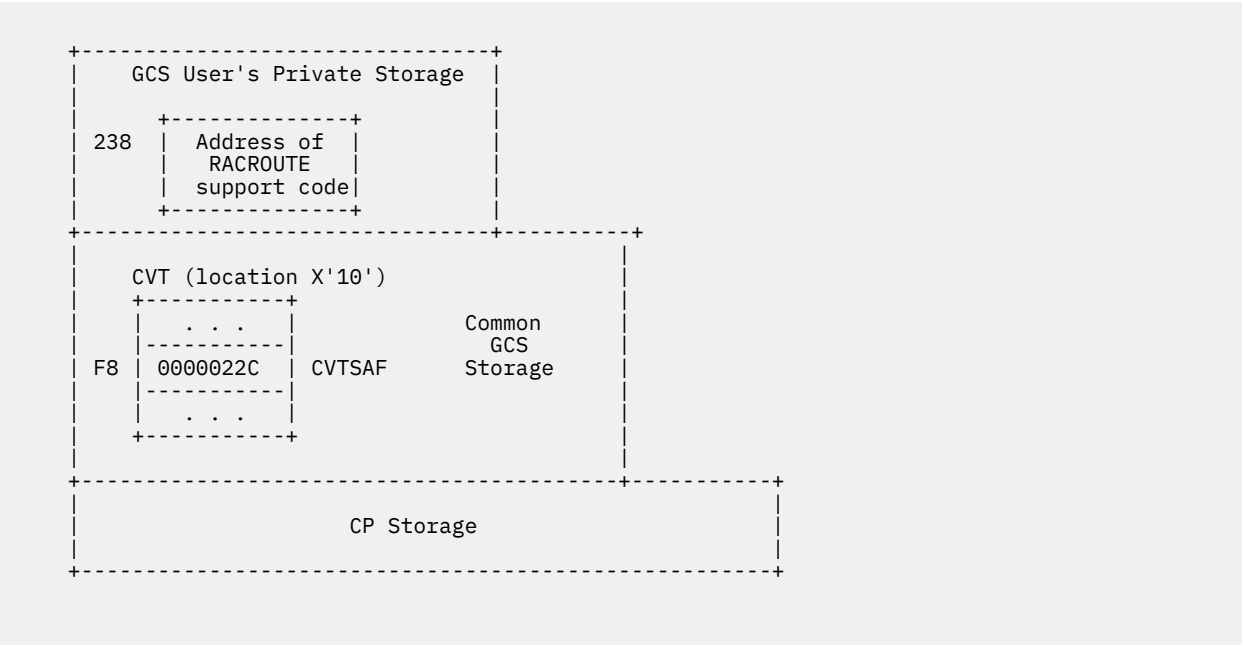
- For the RPIUGCS INIT function:
  - RPIUGCS must place the entry-point address of the RACROUTE support code in a field within the GCS nucleus of the service machine, in the following way:

1. In common GCS storage, look at the contents of the fullword at displacement 16 (X'10') in virtual storage. That fullword contains the address of a control block known as CVTSECT, or CVT (communications vector table).
2. Add 248 (X'F8') to the address of the CVT. The fullword at that address within the CVT, named CVTSAF, contains zeros. Replace those zeros with the following address: 0000022C.
3. Add 12 (X'0C') to the address stored at CVTSAF (X'0000022C'). At this final address (X'00000238') in the GCS virtual machine, store the entry-point address of the RACROUTE support code. This entry point gets control when a RACROUTE request is made.

– RPIUGCS should do whatever processing is necessary to initialize the remainder of the RACROUTE support code.
– If initialization of the RACROUTE support code is successful, the RPIUGCS INIT function should return a zero return code.

The following diagram illustrates the steps listed above:

Establishing a RACROUTE Environment in GCS



If a RACROUTE request is issued and this environment has not been set up, the CVTSAF field is zero, and the RACROUTE macro ends with return code 4.

- For the RPIUGCS TERM function:
  – RPIUGCS should set the CVTSAF field in common GCS storage to zero.
  – RPIUGCS should set the address at X'238' in the GCS virtual machine to zero.
  – RPIUGCS should do whatever cleanup processing the external security manager needs.

Providing RACROUTE Support Code

When the RACROUTE macro expansion is executed, control is passed (through BALR) to the RACROUTE support code.

The following general requirements apply for the RACROUTE support code:

Authorization:
Supervisor state or problem state, in any PSW key

Amode:
24-bit or 31-bit (same as caller of RACROUTE request)

Linkage Conventions:
OS standard

Registers
Meaning

0
Undefined

1
Address of SAFP

2–12
Undefined

13
Save-area address

14
Return address

15
Entry-point address

Entry

On entry to the RACROUTE support code, register 1 contains the address of the RACROUTE parameter list (mapped by macro ICHSAFP). The field SAFPRACP in the RACROUTE parameter list contains the offset from the base address of ICHSAFP to the request-specific portion of the parameter list. The format of the request-specific portions varies, depending on the REQUEST= parameter coded on the RACROUTE invocation. The following table shows which request-specific parameter list is assigned to each REQUEST= keyword:

Table 13. RACROUTE Macro Keywords and Their Request-Specific Parameter Lists

RACROUTE REQUEST Type	Request-Specific Parameter List	Location in the Book
REQUEST=AUDIT	AUL	“AUL” on page 362
REQUEST=AUTH	ACHKL	“ACHKL” on page 356
REQUEST=DEFINE	RDDFL	“RDDFL” on page 397
REQUEST=DIRAUTH	DAUT	“DAUT” on page 367
REQUEST=EXTRACT	RXTL	“RXTL” on page 425
REQUEST=FASTAUTH	FAST	“FAST” on page 368
REQUEST=LIST	RLST	“RLST” on page 412
REQUEST=STAT	STAT	“STAT” on page 441
REQUEST=TOKENBLD	RIPL	“RIPL” on page 404
REQUEST=TOKENMAP	TSRV	“TSRV” on page 442
REQUEST=TOKENXTR	TSRV	“TSRV” on page 442
REQUEST=VERIFY	RIPL	“RIPL” on page 404
REQUEST=VERIFYX	RIPL	“RIPL” on page 404

See Appendix F, “Data Areas for RACROUTE,” on page 347 for details of the format of these parameter lists.

Exit

After exit from the RACROUTE support code:

- Register 15 and field SAFPSFRC in the RACROUTE parameter list (ICHSAFP) must contain one of the following return codes:

  Table 14. Return Codes for Register 15 and the SAFPSFRC Field in the RACROUTE Parameter List

Hex	(Decimal)	Meaning
0	(0)	The requested function completed successfully.
4	(4)	The requested function was not processed.
8	(8)	The requested function was processed, and failed.

- Register 0 and the field SAFPSFRS in the RACROUTE parameter list (ICHSAFP) MAY contain a reason code.
- The RACROUTE parameter list (ICHSAFP) contains the function return code in field SAFPRRET and the function reason code in field SAFPRREA for the requested function. These request-specific return and reason codes are found throughout this book.
- If the ECB1= or ECB2= keyword is specified on a RACROUTE invocation, the RACROUTE support code is expected to post the ECBs when the request is complete.
- Register 1 and the field SAFPRETD in the RACROUTE parameter list (ICHSAFP) may contain the address of a data area returned by the RACROUTE invocation. See each specific RACROUTE request type for details.

Appendix D. Requesting Security Services

The following section contains guidance information for application programmers requesting the services of the security product.

Application writers who need to use the services of a security product should try to write their programs in a way that is compatible with RACF. In turn, the developer of a security product other than RACF should provide function that is compatible with RACF. In this way, application programs should function with RACF or with some other external security product.

Since the functions of RACF are enhanced in new releases, an application may require a specific minimum level of RACF support in order to work correctly. If this is true, you should document this minimum level so that users of your application who have RACF installed will know what level they need. Also, other security-product implementers will then know what level of function they must provide to work correctly with your application. And users of other security products will know what level they need to install to run your application.

Your application should use RACROUTE rather than the older macros RACHECK, RACDEF, RACLIST, RACXTRT, RACINIT, FRACHECK, and RACSTAT. Note that on z/VM, the older macros can only be used within the RACF service machine itself, and not from another virtual machine.

Some programming interfaces to the security product may not be supported by all RACF-compatible security products. For example, the ICHEINTY interface (described in *z/VM: RACF Security Server Macros and Interfaces*) is an interface that is only supported by RACF. If you want your application to work for users of RACF and of other security products, you should not make use of interfaces that are RACF-specific. The RACF-specific interfaces documented in this book are:

- The ACEEFCGP pointer to the list-of-groups table in the ACEE control block.
- The RACROUTE REQUEST=EXTRACT,TYPE=EXTRACT or TYPE=REPLACE functions that specify FIELDS= with a segment name (specified or implied) of BASE. You should not assume that any security product except RACF supports the extraction or replacement of named field information in the base segment of a profile.

  The other security products, however, should support extraction or replacement of data in the other segments, such as DFP and TSO, if they are to be compatible with RACF. They should also support all the other functions of RACROUTE REQUEST=EXTRACT, for RACF compatibility.

As long as you avoid the RACF-specific interfaces, your program should work with any RACF-compatible security product. If you receive a SAF return code of 4 (in SAFPSFRC) with a return code and reason code of 0 in SAFPRRET and SAFPRREA, this indicates that there is no security product, or the product does not support the request you are making. As an application designer, you are then free to make your own decision to continue with your processing, or to terminate processing.

© Copyright IBM Corp. 1990, 2022

Appendix E. Sample RACROUTE Program for Shared User IDs

This appendix contains a sample RACROUTE program for shared user IDs.

This excerpt from a sample (non-reentrant) assembler program issues Diagnose 26C subcode 4 to determine the surrogate user for the shared user ID DIRMAINT. If a surrogate user ID exists, RACF creates an ACEE for DIRMAINT containing information about the surrogate user ID. This ACEE is supplied in a subsequent RACROUTE REQUEST=AUTH to determine if DIRMAINT has READ access to USER1's 191 disk.

```
RACROUTE CSECT
                                                      SPACE
         STM   R14,R12,12(R13)    *********************************
         BALR  R12,0              *                               *
         USING *,R12              *    STANDARD                   *
         ST    R13,SAVE+4         *            ENTRY              *
         LA    R15,SAVE           *                LINKAGE        *
         ST    R15,8(0,R13)       *                               *
         LR    R13,R15            *********************************
                                                      SPACE
*********************************************************************
* Issue DIAGNOSE 26C to find the user ID that is logged onto DIRMAINT.*
*********************************************************************
         LA    R4,X'04'       RY (subcode)
         LA    R2,SHAREDU     RX (addr of input)
         LA    R3,BYUSER      RX+1 (addr of output)
         DC    X'8324026C'    Execute the DIAGNOSE instruction
         LTR   R5,R5          RY+1 = return code
         BNZ   EXIT           non-zero, no BYUSER returned
         CLC   SHAREDU,BYUSER Did user LOGON BY to himself?
         BE    EXIT           No BYUSER
         MVC   RACSHRN,BYUSER Move BY user ID to RACROUTE area
*
*********************************************************************
* Issue RACROUTE REQUEST=VERIFY to create an ACEE for DIRMAINT       *
* identifying the surrogate user ID determined from the DIAGNOSE 26C. *
*********************************************************************
                                                      SPACE
         RACROUTE REQUEST=VERIFY,                                    X
               RELEASE=1.9.2,                                        X
               WORKA=RACWORKA,                                       X
               USERID=SHAREDL,                                       X
               SUSERID=RACSHRL,                                      X
               PASSCHK=NO,                                           X
               ACEE=ACEEADR,                                         X
               ENVIR=CREATE,                                         X
               MF=S
                                                      SPACE

*********************************************************************
* Issue RACROUTE REQUEST=AUTH to determine if DIRMAINT has READ      *
* access to USER1's 191 disk. All audit records created by RACF      *
* contain both DIRMAINT and the user ID that is currently            *
* logged on to DIRMAINT.                                             *
*********************************************************************
         L        R6,ACEEADR  R6 --> acee pointer
*
                                                      SPACE
         RACROUTE REQUEST=AUTH,                                      X
               CLASS=CLASDISK,                                       X
               ACEE=(R6),                                            X
               ENTITYX=DISKNAME,                                     X
               ATTR=READ,                                            X
               WORKA=RACWORKA,                                       X
               RELEASE=1.9.2,                                        X
               MF=S
                                                      SPACE
*
EXIT     DS    0H                 RETURN TO CALLER
*
         L     R13,SAVE+4         *********************************
```

© Copyright IBM Corp. 1990, 2022

```
         L     R14,12(R13)        *   STANDARD EXIT LINKAGE      *
         LM    R0,R12,20(R13)     *                              *
         BR    R14                ********************************
                                                 EJECT
*********************************************************************
* Variable declarations follow.                                     *
*********************************************************************
ACEEADR  DS    F              Area for VERIFY to return ACEE addr
*
RACWORKA DS    CL512          Racroute work area
*
CLASDISK DC    AL1(7),CL8'VMMDISK' VMMDISK class
*
DISKNAME DS    0H             Resource being checked
BUFLEN   DC    H'22'          Maximum buffer length
ACTLEN   DS    H              Actual buffer length
PROFNAME DC    CL22'USER1.191' PROFILE NAME
*
         DS    0D             Align to doubleword boundary
SHAREDU  DC    CL8'DIRMAINT'  Input for Diagnose 26C
BYUSER   DC    CL8'       '   Output for Diagnose 26C
*
RACSHRL  DC    XL1'08'        LENGTH OF SURROGATE USER FOR RACROUTE
RACSHRN  DC    CL8' '         User ID name for RACROUTE
*
SHAREDL  DC    XL1'08'        Length of SHARED user ID for RACROUTE
SHAREDN  DC    CL8'DIRMAINT'  User ID name for RACROUTE
SAVE     DC    18F'0'             REGISTER SAVE AREA
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

Appendix F. Data Areas for RACROUTE

This section contains graphic presentations of data areas used by RACF and application programs. This information includes data areas that are intended to be used as programming interface and information about data areas that are not intended to be used as programming interfaces.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of RACF. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Unless otherwise specified, for data areas classified as programming interfaces, the **MACRO ID** in the header is part of the programming interface. **ALL** other header information is included for diagnostic purposes **ONLY**.

A *data area name* that is designated as part of the programming interface differentiation is one of the following:

- Macro ID
- DSECT name
- Commonly-used name

Before including the *data area name* in a program, refer to the data area header for the applicable **Macro ID**

When an entire data area is classified as a programming interface, “RESERVED FOR USER” or “RESERVED FOR INSTALLATION” fields are part of the interface; all other **“RESERVED ...”** fields are **NOT** part of the interface.

If only certain fields in a data area are intended or not intended for use as a programming interface, the specific field name(s) are differentiated within this appendix.

For a field that is part of the programming interface, the only information that is part of the interface for writing programs is:

- Field name
- Data type
- Field length
- Description (purpose or allowed values)

ACEE

The following fields:

NOT programming interface information
ACEEAMP ACEEMDLS ACEECGRP ACEECLCP ACEEGATA ACEEPADS ACEE3PTY

© Copyright IBM Corp. 1990, 2022

ACEEOCOX
ACEEPTDS

End of NOT programming interface information

Common Name:	Accessor Environment Element (ACEE)
Macro ID:	IHAACEE
DSECT Name:	ACEE
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	ACEE Offset: 0 Length: 4
Subpool and Key:	Subpool 255 and key 0, or subpool specified by issuer of RACROUTE REQUEST=VERIFY (may reside above 16M)
Size:	168 bytes. Does not include any data pointed to by ACEE.
Created by:	SAF (MVS™) or RACF* depending on the parameters specified on RACROUTE REQUEST=VERIFY
Pointed to by:	ASXBSENV or TCBSENV (MVS only), or a field supplied by issuer of RACROUTE REQUEST=VERIFY. (ACEEs pointed to by ASXBSENV or TCBSENV always reside below 16M).
Serialization:	None

Function:

This mapping macro maps the ACEE. This control block represents the authorities of a single accessor in the address space.

Note:

1. If you use ACEEIEP, it must point to an area of storage you obtained by using a GETMAIN. RACF will free this area when it frees the ACEE. For RACF to do this, the first word of the area must contain both the subpool and the length of the area. (The subpool appears in the high order byte, and the length appears in the next three bytes.)

   If you do not conform to this requirement in your usage of ACEEIEP, you must supply a RACINIT exit to free the area and set the ACEEIEP field to zero when a caller issues a RACINIT DELETE. In certain situations, however, your exit will not be called during RACF error recovery, and unpredictable results may occur. Therefore, IBM strongly recommends that you adhere to the specified requirements.

   Following are some examples of nonconforming use of ACEEIEP.

   a. ACEEIEP contains data, rather than a pointer.
   b. ACEEIEP contains a pointer, however the first word of the area pointed to by ACEEIEP does not contain the subpool and length information for the area.
   c. ACEEIEP contains a pointer, and the first word of the area pointed to contains the subpool and length information for a data area that points to additional area obtained using GETMAIN.

      Note: This situation would not necessarily cause an abend, but would result in a failure to FREE the acquired data area.

   In addition, if your usage of ACEEIEP does not conform to the specified requirements, or if your data area contains any pointers to other data areas, you will have to provide an ACEE compression/expansion exit. See *z/VM: RACF Security Server System Programmer's Guide* for additional details.
2. Within an IMS address space, ACEEAPTR is reserved for use by IMS when bringing up IMS or signing on to IMS.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	168	ACEE	ACCESSOR ENVIRONMENT ELEMENT
0	(0)	CHARACTER	4	ACEEACEE	ACRONYM IN EBCDIC -ACEE-
4	(4)	ADDRESS	4	ACEECORE	ACEE SUBPOOL AND LENGTH
4	(4)	ADDRESS	1	ACEESP	ACEE SUBPOOL NUMBER
5	(5)	ADDRESS	3	ACEELEN	LENGTH OF ACEE
8	(8)	UNSIGNED	1	ACEEVRSN	VERSION = 1.
9	(9)	CHARACTER	3	*	RESERVED

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
12	(C)	ADDRESS	4	ACEEIEP	RESERVED FOR INSTALLATION. IF USED, IT MUST POINT TO A ONE BYTE SUBPOOL FOLLOWED BY A THREE BYTE LENGTH.
16	(10)	ADDRESS	4	ACEEINST	ADDRESS OF INSTALLATION SUPPLIED USER DATA - FROM USER ENTRY
20	(14)	CHARACTER	9	ACEEUSER	USERID INFORMATION
20	(14)	ADDRESS	1	ACEEUSRL	USERID LENGTH
21	(15)	CHARACTER	8	ACEEUSRI	USERID OR , IF NOT SUPPLIED AT RACINIT
29	(1D)	CHARACTER	9	ACEEGRP	GROUP NAME INFORMATION
29	(1D)	ADDRESS	1	ACEEGRPL	GROUP NAME LENGTH
30	(1E)	CHARACTER	8	ACEEGRPN	CONNECT GROUP NAME
38	(26)	BITSTRING	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..`		ACEEROA	1 - Read-only auditor attribute
		`.... ..1.`		ACEEPRIV	1 - USER IS A STARTED PROCEDURE WITH THE PRIVILEGED ATTRIBUTE
		`.... ...1`		ACEERACF	1 - RACF DEFINED USER
39	(27)	BITSTRING	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

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		.... 1...		*	RESERVED FOR COMPATIBILITY
		.... .1..		*	RESERVED
		.... ..1.		*	RESERVED
		1		ACEENONE	1 - NO AUTHORITY TO RESOURCE
40	(28)	BITSTRING	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)	CHARACTER	3	ACEEDATE	DATE OF RACINIT
44	(2C)	CHARACTER	8	ACEEPROC	NAME OF STARTED PROC OR BLANKS IF NOT STARTED PROC
52	(34)	ADDRESS	4	ACEETRMP	ADDRESS OF TERMINAL RELATED INFORMATION. ZERO FOR NON- TERMINAL USERS
56	(38)	BITSTRING	2	ACEEFLG4	MISCELLANEOUS FLAGS 2
		1...		*	RESERVED
		.1..		*	RESERVED
		..1.		ACEEUATH	1 - USER IS AUTHORIZED TO DEFINE OTHER USERS

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		...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
56	(38)	BITSTRING	1	*	RESERVED
58	(3A)	ADDRESS	1	ACEEAPLV	APPLICATION LEVEL NUMBER
59	(3B)	ADDRESS	1	ACEETRLV	TERMINAL LEVEL NUMBER
60	(3C)	ADDRESS	4	ACEETRDA	ADDRESS OF INSTALLATION SUPPLIED DATA FROM TERMINAL ENTRY
64	(40)	CHARACTER	8	ACEETRID	TERMINAL ID
72	(48)	ADDRESS	4	ACEEAMP	ADDRESS FIRST ANCHORED MODEL
76	(4C)	BITSTRING	4	ACEECLTH	USER CLASS AUTHORIZATIONS - THESE BIT POSITIONS ARE MAPPED BY THE CLASS DESCRIPTOR ENTRIES ANCHORED OFF THE RACF CVT
80	(50)	ADDRESS	4	ACEECLCP	ANCHOR FOR INSTORAGE PROFILE TREES BUILT BY THE RACLIST FUNCTION
84	(54)	ADDRESS	4	ACEEAPTR	ADDRESS FIELD RESERVED FOR APPLICATION USAGE
88	(58)	CHARACTER	8	ACEEAPLN	NAME OF APPLICATION TO WHICH USER IS CONNECTED OR BLANKS IF NO APPLICATION SPECIFIED
96	(60)	ADDRESS	4	ACEEAPDA	ADDRESS INSTALLATION SUPPLIED DATA FROM APPLICATION ENTRY
100	(64)	ADDRESS	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.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
104	(68)	ADDRESS	4	ACEEMDLS	ADDRESS OF THE DATA SET MODEL NAME ARRAY. ZERO, IF ARRAY NOT OBTAINED BY RACINT.
108	(6C)	ADDRESS	4	ACEECGRP	ADDRESS OF TABLE CONTAINING THE LIST OF GROUPS THIS USERID IS A MEMBER OF.
112	(70)	ADDRESS	4	ACEEGATA	ADDRESS OF THE GENERIC ANCHOR TABLE
116	(74)	ADDRESS	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)	ADDRESS	4	ACEEDSLP	ADDRESS OF THE LIST OF CATEGORIES TO WHICH THIS USER IS ALLOWED ACCESS
124	(7C)	ADDRESS	4	ACEEDAT4	4 BYTE DATE FIELD IN THE FORM OF ccyydddF WHERE cc DENOTES THE CENTURY X'00' FOR 1900 AND X'01' FOR 2000.
128	(80)	ADDRESS	4	ACEEPADS	ADDRESS OF THE LIST OF DATA SETS ACCESSED BY CONTROLLED PROGRAMS EXECUTED BY THIS USER
132	(84)	UNSIGNED	1	ACEESLVL	MAXIMUM SECURITY LEVEL ACCESSED BY THIS USER
133	(85)	BITSTRING	1	ACEEFLG5	MISCELLANEOUS FLAGS
		`1... ....`		ACEEMODE	1 - ACEE MODE IS IN 31-BIT MODE
		`.1.. ....`		ACEED4OK	1 - ACEEDAT4 CONTAINS DATA
					0 - ACEEDAT4 NOT USED
		`..11 1111`		*	RESERVED
134	(86)	CHARACTER	1	ACEEFLG6	MISCELLANEOUS FLAGS
		`1... ....`		ACEEMFAE	1 - USER IS MFA ENABLED
		`.1.. ....`		ACEEMFAF	1 - USER IS MFA FALLBACK ENABLED

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		..1.		ACEEMFAA	1 - USER AUTHENTICATED WITH MFA
136	(88)	ADDRESS	4	ACEE3PTY	ADDRESS OF ACEE CREATED BY THIRD PARTY RACHECK SVC PROCESSING
140	(8C)	ADDRESS	4	ACEEPLCL	POINTER TO EXTENDED CLASS AUTHORIZATION MASK, OR 0
144	(90)	CHARACTER	8	ACEESUID	SURROGATE USERID (AUDIT)
152	(98)	ADDRESS	4	ACEEOCOX	POINTER TO OCO EXTENSION
156	(9C)	ADDRESS	4	ACEEPTDS	POINTER TO FIRST TDS TABLE
160	(A0)	CHARACTER	4	*	RESERVE THESE 4 BYTES SO ACEE WILL HAVE THE LENGTH OF THE MULTITLE OF 8.
164	(A4)	ADDRESS	4	ACEETOKP	POINTER TO ACEETOKN

Constants

Description of constants.

Len	Type	Value	Name	Description
1	DECIMAL	1	ACEEVR01	ACEE VERSION NUMBER = 1.
1	DECIMAL	2	ACEEVR02	ACEE VERSION NUMBER = 2.
1	DECIMAL	2	ACEECURV	ACEE CURRENT VERSION NUMBER

Cross Reference

Name	Hex Offset	Hex Value	Level
ACEE	0		1
ACEEACEE	0		2
ACEEADSP	26	40	3
ACEEALTR	27	80	3
ACEEAMP	48		2
ACEEAPDA	60		2

Name	Hex Offset	Hex Value	Level
ACEEAPLN	58		2
ACEEAPLV	3A		2
ACEEAPTR	54		2
ACEEAUDT	26	10	3
ACEECGRP	6C		2
ACEECLCP	50		2
ACEECLTH	4C		2
ACEECNTL	27	40	3
ACEECORE	4		2
ACEEDASD	38	08	3
ACEEDATE	29		2
ACEEDAT4	7C		2
ACEEDSLP	78		2
ACEED4OK	85	40' '3	
ACEEFCGP	74		2
ACEEFLG1	26		2
ACEEFLG2	27		2
ACEEFLG3	28		2
ACEEFLG4	38		2
ACEEFLG5	85		2
ACEEGATA	70		2
ACEEGRP	1D		2
ACEEGRPA	28	80	3
ACEEGRPL	1D		3
ACEEGRPN	1E		3
ACEEIEP	C		2
ACEEINST	10		2
ACEELEN	5		3
ACEELOGU	26	08	3
ACEEMDLS	68		2
ACEEMODE	85	80	3
ACEENONE	27	01	3
ACEEOCOX	98		2
ACEEOPER	26	20	3
ACEEPADS	80		2
ACEEPLCL	8C		2

Name	Hex Offset	Hex Value	Level
ACEEPRIV	26	02	3
ACEEPROC	2C		2
ACEEPTDS	9C		2
ACEERACF	26	01	3
ACEEREAD	27	10	3
ACEESLVL	84		2
ACEESP	4		3
ACEESPEC	26	80	3
ACEESUID	90		2
ACEETAPE	38	04	3
ACEETERM	38	02	3
ACEETOKP	A4		2
ACEETRDA	3C		2
ACEETRID	40		2
ACEETRLV	3B		2
ACEETRMP	34		2
ACEEUATH	38	20	3
ACEEUNAM	64		2
ACEEUPDT	27	20	3
ACEEUSER	14		2
ACEEUSRI	15		3
ACEEUSRL	14		3
ACEEVRSN	8		2
ACEE3PTY	88		2

ACHKL

Common Name:	Request-specific portion of the RACROUTE REQUEST=AUTH parameter list
Macro ID:	ICHACHKL
DSECT Name:	ACHKLIST
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Caller-determined Key: Caller-determined Residency: Caller-determined
Size:	Varies with RELEASE= parameter specified
Created by:	RACROUTE REQUEST=AUTH macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP

Serialization: None

Function: Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=AUTH routine.

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	32	ACHKLIST	RACHECK PARAMETER LIST
0	(0)	ADDRESS	4	ACHKINSW	ADDRESS INSTALLATION DATA
0	(0)	UNSIGNED	1	ACHKLENG	LENGTH OF PARAMETER LIST
1	(1)	ADDRESS	3	ACHKINST	ADDRESS INSTALLATION DATA
4	(4)	SIGNED	4	ACHKENTW	ENTITY ADDRESS WORD
4	(4)	UNSIGNED	1	ACHKFLG1	FIRST FLAGS BYTE
		1...		ACHKRFI	RACFIND PARAMETER GIVEN
		.1..		ACHKRFIY	RACFIND=YES
		..1.		ACHKENX	ENTITYX IS SPECIFIED
		...1		ACHKDSTV	DSTYPE=V
		.... 1...		ACHK31IN	31-BIT-ADDRESS LIST INDICATOR
		.... .11.		ACHKLOGS	LOG=NOSTAT (BOTH ON)
		.... .1..		ACHKLOGF	LOG=NOFAIL
		.... ..1.		ACHKLOGN	LOG=NONE
		1		ACHKCSA	ENTITY=(ADDR,CSA)
5	(5)	ADDRESS	3	ACHKENT	ENTITY NAME ADDRESS
8	(8)	SIGNED	4	ACHKCLNW	CLASS NAME ADDRESS WORD
8	(8)	UNSIGNED	1	ACHKFLG2	SECOND FLAGS BYTE
		1...		ACHKTALT	ATTR=ALTER
		.111		*	RESERVED
		.... 1...		ACHKTCTL	ATTR=CONTROL

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
		.... .1..		ACHKTUPD	ATTR=UPDATE
		.... ..1.		ACHKTRD	ATTR=READ
		1		*	RESERVED
9	(9)	ADDRESS	3	ACHKCLN	CLASS NAME ADDRESS
12	(C)	SIGNED	4	ACHKVOLW	VOLSER ADDRESS WORD
12	(C)	UNSIGNED	1	ACHKFLG3	THIRD FLAGS BYTE
		1...		ACHKTAPE	DSTYPE=T
		.1..		ACHKMDEL	DSTYPE=M
		..1.		ACHKPRF	PROFILE ADDR GIVEN
		...1		*	RESERVED
		.... 1...		ACHKVOL	VOLSER PARM SPECIFIED
		.... .1..		ACHKGEN	GENERIC=YES
		.... ..1.		ACHKPRI	PRIVATE=YES
		1		*	RESERVED
13	(D)	ADDRESS	3	ACHKVOLS	VOLSER ADDRESS
16	(10)	ADDRESS	4	ACHKOVOL	OLD VOLSER ADDRESS
20	(14)	ADDRESS	4	ACHKAPPL	APPL NAME ADDRESS
24	(18)	ADDRESS	4	ACHKACEE	ACEE ADDRESS
28	(1C)	ADDRESS	4	ACHKOWNR	OWNER ADDRESS.
32	(20)	CHARACTER		ACHKEND	END OF V1.4 LIST

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
32	(20)	STRUCTURE	16	ACHK31	31-BIT-ADDRESS SAF EXTENSION
32	(20)	ADDRESS	4	ACHKIN31	31-BIT INSTALLATION DATA ADDRESS

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
36	(24)	ADDRESS	4	ACHKENTX	ENTITYX NAME ADDRESS
36	(24)	ADDRESS	4	ACHKEN31	ENTITY NAME/RESOURCE PROFILE ADDRESS
40	(28)	ADDRESS	4	ACHKCL31	CLASS NAME ADDRESS
44	(2C)	ADDRESS	4	ACHKVS31	VOLSER ADDRESS
48	(30)	CHARACTER		ACHK31EN	END OF SAF EXTENSION

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
16	(10)	STRUCTURE	8	ACHK15	RACF 1.5 EXTENSION
16	(10)	ADDRESS	4	ACHKACC1	ACCLVL ADDRESS (1ST PART)
20	(14)	ADDRESS	4	ACHKACC2	ACCLVL ADDRESS (2ND PART)
24	(18)	CHARACTER		ACHK15EN	END OF V1.5 EXTENSION

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
8	(8)	STRUCTURE	4	ACHK17	RACF 1.7 EXTENSION
8	(8)	UNSIGNED	2	ACHKFSEQ	FILE SEQUENCE NO
10	(A)	UNSIGNED	1	ACHKFLGT	TAPE FLAG BYTE
		11..		ACHKTLBL	TAPELBL SPECIFIED B'00'=STD B'10'=BLP B'01'=NL
		..11 1111		*	RESERVED
11	(B)	UNSIGNED	1	ACHKFLG4	FOURTH FLAG BYTE
		1...		ACHKEOS	STATUS=ERASE SPECIFIED
		.1..		ACHKEVD	STATUS=EVERDOM SPECIFIED
		..1.		ACHKWRON	STATUS=WRITEONLY SPECIFIED
		...1		ACHKACCS	STATUS=ACCESS SPECIFIED
		 1111		*	RESERVED
12	(C)	CHARACTER		ACHK17EN	END OF RACF 1.7 EXTENSION

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
4	(4)	STRUCTURE	8	ACHK18	RACF 1.8 EXTENSION
4	(4)	ADDRESS	4	ACHKUSID	USERID POINTER
8	(8)	ADDRESS	4	ACHKGPID	GROUPID POINTER
12	(C)	CHARACTER		ACHK18EN	END OF 1.8 EXTENSION

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
8	(8)	STRUCTURE	4	ACHK18X	RACF 1.8X EXT
8	(8)	ADDRESS	4	ACHKDDPR	DDNAME POINTER
12	(C)	CHARACTER		ACHK8XEN	END OF 1.8X EXT

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
4	(4)	STRUCTURE	20	ACHK19	RACF 1.9 EXTENSION
4	(4)	ADDRESS	4	*	RESERVED
8	(8)	ADDRESS	4	ACHKUTOK	UTOKEN POINTER
12	(C)	ADDRESS	4	ACHKRTOK	RTOKEN POINTER
16	(10)	ADDRESS	4	ACHKLSTR	LOGSTR POINTER
20	(14)	ADDRESS	4	ACHKRCVR	RECVR POINTER
24	(18)	CHARACTER		ACHK19EN	END OF 1.9 EXTENSION

Cross Reference

Name	Hex Offset	Hex Value	Level
ACHKACCS	B	10	3
ACHKACC1	10		2
ACHKACC2	14		2
ACHKACEE	18		2
ACHKAPPL	14		2
ACHKCLN	9		3
ACHKCLNW	8		2
ACHKCL31	28		2
ACHKCSA	4	01	4

Name	Hex Offset	Hex Value	Level
ACHKDDPR	8		2
ACHKDSTV	4	10	4
ACHKEND	20		2
ACHKENT	5		3
ACHKENTW	4		2
ACHKENTX	24		2
ACHKENX	4	20	4
ACHKEN31	24		3
ACHKEOS	B	80	3
ACHKEVD	B	40	3
ACHKFLGT	A		2
ACHKFLG1	4		3
ACHKFLG2	8		3
ACHKFLG3	C		3
ACHKFLG4	B		2
ACHKFSEQ	8		2
ACHKGEN	C	04	4
ACHKGPID	8		2
ACHKINST	1		3
ACHKINSW	0		2
ACHKIN31	20		2
ACHKLENG	0		3
ACHKLIST	0		1
ACHKLOGF	4	04	5
ACHKLOGN	4	02	5
ACHKLOGS	4	04	4
ACHKLSTR	10		2
ACHKMDEL	C	40	4
ACHKOVOL	10		2
ACHKOWNR	1C		2
ACHKPRF	C	20	4
ACHKPRI	C	02	4
ACHKRCVR	14		2
ACHKRFI	4	80	4
ACHKRFIY	4	40	4
ACHKRTOK	C		2

Name	Hex Offset	Hex Value	Level
ACHKTALT	8	80	4
ACHKTAPE	C	80	4
ACHKTCTL	8	08	4
ACHKTLBL	A	80	3
ACHKTRD	8	02	4
ACHKTUPD	8	04	4
ACHKUSID	4		2
ACHKUTOK	8		2
ACHKVOL	C	08	4
ACHKVOLS	D		3
ACHKVOLW	C		2
ACHKVS31	2C		2
ACHKWRON	B	20	3
ACHK15	10		1
ACHK15EN	18		2
ACHK17	8		1
ACHK17EN	C		2
ACHK18	4		1
ACHK18EN	C		2
ACHK18X	8		1
ACHK19	4		1
ACHK19EN	18		2
ACHK31	20		1
ACHK31EN	30		2
ACHK31IN	4	08	4
ACHK8XEN	C		2

AUL

Common Name:	Request-specific portion of the RACROUTE REQUEST=AUDIT parameter list
Macro ID:	ICHPAUL
DSECT Name:	AUDLIST
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	36 bytes

Created by:	RACROUTE REQUEST=AUDIT macro
Pointed to by:	Address of SAFP plus offset at SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=AUDIT routine.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	48	AUDLIST	RACAUDIT parameter list
0	(0)	UNSIGNED	2	AUDVERS	Parameter list version
2	(2)	UNSIGNED	2	AUDLEN	Parameter list length
4	(4)	ADDRESS	4	AUDEVENT	Address of event name
8	(8)	UNSIGNED	2	AUDEQUAL	Event code qualifier
10	(A)	UNSIGNED	2	*	Reserved
12	(C)	ADDRESS	4	AUDCLASS	Address of class name
16	(10)	ADDRESS	4	AUDENTYX	Address of entity name
20	(14)	ADDRESS	4	AUDACEE	Address of ACEE
24	(18)	ADDRESS	4	AUDLOGST	Address of LOGSTR data
28	(1C)	UNSIGNED	1	AUDRESUL	Result byte
29	(1D)	UNSIGNED	3	*	Reserved
32	(20)	ADDRESS	4	* (4)	Reserved

Cross Reference

Name	Hex Offset	Hex Value	Level
AUDACEE	14		2
AUDCLASS	C		2
AUDENTYX	10		2
AUDEQUAL	8		2
AUDEVENT	4		2
AUDLEN	2		2
AUDLIST	0		1
AUDLOGST	18		2
AUDRESUL	1C		2
AUDVERS	0		2

CGRP

NOT programming interface information

- Field CGRPGPAT
- When addressed via ACEECGRP, the CGRP data area is not intended for customer use as programming interface information.

End of NOT programming interface information

Common Name:	Connect Group Name Table Definition
Macro ID:	ICHPCGRP
DSECT Name:	CGRP, CGRPENTD
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	CGRP Offset: 0 Length: 4
Subpool and Key:	255 and key 0, or subpool specified by issuer of RACROUTE REQUEST=VERIFY (may reside above 16M)
Size:	Variable, Fixed 32 bytes + 24 bytes per connect group
Created by:	Various RACF functions
Pointed to by:	ACEECGRP or ACEEFCGP field of the ACEE data area
Serialization:	NONE WHEN POINTED TO BY ACEEFCGP
Function:	This table contains the names of the groups that the ACEEUSRI userid is a member of.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	CGRP	CONNECT GROUP NAME TABLE.
0	(0)	CHARACTER	32	CGRPHADR	CGRP header
0	(0)	CHARACTER	4	CGRPID	TABLE ID.
4	(4)	CHARACTER	4	CGRPCORE	CGRP SUBPOOL AND LENGTH.
4	(4)	UNSIGNED	1	CGRPSP	SUBPOOL NUMBER.
5	(5)	ADDRESS	3	CGRPLEN	LENGTH OF CGRP.
8	(8)	SIGNED	2	CGRPNUM	MAXIMUM ENTRIES IN TABLE.
10	(A)	UNSIGNED	1	CGRPVRSN	VERSION = 1.
11	(B)	CHARACTER	1	*	RESERVED.
12	(C)	SIGNED	4	CGRPSYNC	SYNCHRONIZE VALUE.
16	(10)	ADDRESS	4	CGRPGPAT	ADDRESS OF GROUP AUTHORITIES TABLE, OR ZERO IF NO SUCH TABLE EXISTS
20	(14)	CHARACTER	12	*	RESERVED

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
32	(20)	CHARACTER	24	CGRPENT (*)	GROUP NAME ENTRY.
32	(20)	CHARACTER	8	CGRPNAME	GROUP NAME.
40	(28)	BITSTRING	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
41	(29)	BITSTRING	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)	SIGNED	2	CGRPGPNM	NUMBER OF ENTRIES IN GROUP AUTHORITY TABLE RELATED TO THIS CONNECT GROUP
44	(2C)	SIGNED	4	CGRPGPTE	ADDRESS OF FIRST GROUP AUTHORITY TABLE ENTRY RELATED TO THIS CONNECT GROUP

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
48	(30)	SIGNED	2	CGRPSUPG	INDEX IN CGRPENT OF ENTRY FOR SUPERIOR GROUP OF THIS ENTRY, TO WHICH THE USER IS CONNECTED
50	(32)	CHARACTER	6	*	RESERVED.
56	(38)	CHARACTER		*	END OF ENTRY.

Constants

Len	Type	Value	Name	Description
4	CHARACTER	CGRP	CGRPTID	TABLE ID.

Cross Reference

Name	Hex Offset	Hex Value	Level
CGRP	0		1
CGRPAUDT	29	10	4
CGRPAUTH	29		3
CGRPCHK	28	80	4
CGRPCOMP	28	20	4
CGRPCORE	4		3
CGRPENT	20		2
CGRPGPAT	10		3
CGRPGPNM	2A		3
CGRPGPTE	2C		3
CGRPHADR	0		2
CGRPID	0		3
CGRPIND	28		3
CGRPLEN	5		4
CGRPNAME	20		3
CGRPNUM	8		3
CGRPOPER	29	20	4
CGRPPROP	28	10	4
CGRPREFR	28	40	4
CGRPSP	4		4
CGRPSPEC	29	80	4

Name	Hex Offset	Hex Value	Level
CGRPSUPG	30		3
CGRPSYNC	C		3
CGRPVRSN	A		3

DAUT

Common Name:	Request-specific portion of the RACROUTE REQUEST=DIRAUTH parameter list
Macro ID:	None
DSECT Name:	None
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Variable
Created by:	RACROUTE REQUEST=DIRAUTH macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=DIRAUTH routine.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	8	DAUTPARM	DIRAUTH parameters
0	(0)	BITSTRING	1	DAUTLOGP	Auditing option flags
		1...		DAUTASIS	1 = ASIS
		.1..		DAUTNFAI	1 = NOFAIL
		..11 1111		*	Reserved
1	(1)	CHARACTER	3	*	Reserved
4	(4)	ADDRESS	4	DAUTRTOK	Message RTOKEN address

Cross Reference

Name	Hex Offset	Hex Value	Level
DAUTASIS	0	80	3
DAUTLOGP/entry	0		2
DAUTNFAI	0	40	3

Name	Hex Offset	Hex Value	Level
DAUTPARM	0		1
DAUTRTOK	4		2

FAST

Common Name:	Request-specific portion of the RACROUTE REQUEST=FASTAUTH parameter list
Macro ID:	None
DSECT Name:	None
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Variable
Created by:	RACROUTE REQUEST=FASTAUTH macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=FASTAUTH routine.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	28	FASTPARM	FASTAUTH parameters
0	(0)	BITSTRING	1	FASTATTR	ATTR= Flags
		1...		FASTALTR	1 = ALTER requested
		.111		*	Reserved
		.... 1...		FASTCNTL	1 = CONTROL requested
		.... .1..		FASTUPDT	1 = UPDATE requested
		.... ..1.		FASTREAD	1 = READ requested
		1		*	Reserved
1	(1)	CHARACTER	3	*	Reserved
4	(4)	ADDRESS	4	FASTENTP	Address of entity name
8	(8)	ADDRESS	4	FASTCLAS	Address of class name
12	(C)	ADDRESS	4	FASTACEE	Address of ACEE to use
16	(10)	ADDRESS	4	FASTAPPL	Address of application name
20	(14)	ADDRESS	4	FASTWKA	Address of 16 word workarea

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
24	(18)	ADDRESS	4	FASTINST	Address of installation exit data field

Cross Reference

Name	Hex Offset	Hex Value	Level
FASTACEE	C		2
FASTALTR	0	80	3
FASTAPPL	10		2
FASTATTR	0		2
FASTCLAS	8		2
FASTCNTL	0	08	3
FASTENTP	4		2
FASTINST	18		2
FASTPARM	0		1
FASTREAD	0	02	3
FASTUPDT	0	04	2
FASTWKA	14		2

ISP

NOT programming interface information

The following fields:

- RACRTE
- RACRSE
- RACRNE

End of NOT programming interface information

Common Name:	RACF In-Storage Profile
Macro ID:	ICHPISP
DSECT Name:	RACRTE,RACRSE,RACRNE,RPEINSD, RACRPE,RPEACCLE
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Subpool and Key:	255 or subpool specified by issuer of RACLIST and key 0 (may reside above 16M)

Size: 1st Section: 32 bytes. 2nd Section: 8 bytes plus a variable of unknown length at offset 8. 3rd Section: 16 bytes plus a variable of unknown length at offset 16. 4th Section: 74 bytes. 5th Section: 1 bytes plus a variable of unknown length at offset 1. 6th Section: 1 bytes plus a variable of unknown length at offset 1. 7th Section: 9 bytes per entry in the access list. 8th Section: 2 bytes per category. 9th Section: 10 bytes plus 1 - 8 characters at offset 10. 10th Section: 31 bytes plus a variable of unknown length at offset 31. 11th Section: Variable. 12th Section: 1 byte.

Created by: RACLIST processing

Pointed to by: ACEECLCP field of the ACEE data area. On systems prior to MVS/ESA and on VM systems, also pointed to by CNSTRCLP. Individual profiles can be located in two ways: 1. RACROUTE REQUEST=AUTH with ENTITY=(...,CSA or PRIVATE), which will return a copy of the profile mapped by ICHRRPF or 2. For a RACLIST tree pointed to from the ACEE using RACROUTE REQUEST= FASTAUTH, which will return a pointer to a profile that was used in word 14 of the work area pointed to by WKAREA.

Serialization: None

Function: This table contains profiles for general resources in a class plus control information for locating individual profiles.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	32	RACRTE	RACLIST CLASS TREE ANCHOR ELEMENT
0	(0)	ADDRESS	4	RTENEXT	ADDRESS OF NEXT ANCHOR OR 0
4	(4)	ADDRESS	4	RTECLASS	ADDRESS OF CLASS DESCRIPTOR ENTRY FOR THIS CLASS
8	(8)	ADDRESS	4	RTETREE	ADDRESS OF TOP NODE IN TREE OR 0
12	(C)	ADDRESS	4	RTESTORE	ADDRESS OF STORAGE BLOCK LIST OR 0
16	(10)	CHARACTER	2	RTESPNS	PROFILE & NODE SUBPOOL NUMBERS
16	(10)	UNSIGNED	1	RTEPSPN	SUBPOOL NUMBER FOR PROFILES
17	(11)	UNSIGNED	1	RTENSPN	SUBPOOL NUMBER FOR TREE NODES
18	(12)	UNSIGNED	1	RTEASPN	SUBPOOL NUMBER OF THIS BLOCK
19	(13)	CHARACTER	1	*	RESERVED
20	(14)	ADDRESS	4	RTEGENL	ADDRESS OF GENERIC PROFILE LIST OR 0

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
24	(18)	SIGNED	4	RTESIZE	TOTAL STORAGE USED FOR RACLISTED PROFILES AND NODES
28	(1C)	SIGNED	4	RTEGNUM	TOTAL NUMBER OF GROUPING PROFILES THAT CONTAIN MEMBERS

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	RACRSE	RACLIST CLASS TREE STORAGE BLOCK
0	(0)	ADDRESS	4	RSENEXT	ADDRESS OF NEXT STORAGE BLOCK OR 0
4	(4)	SIGNED	2	RSESIZE	LENGTH OF STORAGE BLOCK
6	(6)	UNSIGNED	1	RSEPOOL	SUBPOOL NUMBER OF STORAGE BLOCK
7	(7)	UNSIGNED	1	*	RESERVED
8	(8)	CHARACTER	*	RSESTORE	USEABLE STORAGE (RSESIZE-4 BYTES)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	RACRNE	RACLIST CLASS TREE NODE ELEMENT
0	(0)	ADDRESS	4	RNELEFT	ADDRESS OF LEFT DAUGHTER NODE OR 0
4	(4)	ADDRESS	4	RNEPROF	ADDRESS OF PROFILE FOR THIS NODE
8	(8)	ADDRESS	4	RNERIGHT	ADDRESS OF RIGHT DAUGHTER NODE OR 0
12	(C)	SIGNED	4	RNEBAL	TREE BALANCING FACTOR DURING TREE CREATION
12	(C)	ADDRESS	4	RNEUP	POINTER TO MOTHER NODE DURING TREE DELETION
16	(10)	CHARACTER	*	RNEKEY	KEY (LENGTH DETERMINED BY MAXIMUM NAME LENGTH FOR CLASS IN THE CLASS DESCRIPTOR ELEMENT)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	78	RACRPE	RESOURCE PROFILE ELEMENT
0	(0)	UNSIGNED	2	RPEPLEN	PHYSICAL STORAGE LENGTH OF BLOCK
2	(2)	UNSIGNED	2	RPELLEN	LOGICAL LENGTH OF BLOCK
4	(4)	UNSIGNED	2	RPEUCNT	NUMBER OF RESOURCES SHARING THIS PROFILE
6	(6)	CHARACTER	4	RPEATTR	ATTRIBUTE FLAGS
6	(6)	BITSTRING	1	RPEUACC	UNIVERSAL ACCESS
7	(7)	BITSTRING	1	RPEAUDIT	AUDIT FLAGS
8	(8)	BITSTRING	1	RPEGAUD	GLOBAL AUDIT FLAGS
9	(9)	BITSTRING	1	RPELEVEL	RESOURCE LEVEL
10	(A)	UNSIGNED	2	RPEACCNO	NUMBER OF ENTRIES IN ACCESS LIST
12	(C)	UNSIGNED	2	RPEACCOF	OFFSET TO ACCESS LIST
14	(E)	UNSIGNED	2	RPEINSOF	OFFSET TO INSTALLATION DATA
16	(10)	UNSIGNED	2	RPEAPPOF	OFFSET TO APPLICATION DATA
18	(12)	CHARACTER	8	RPEOWNER	OWNER OF RESOURCE PROFILE
26	(1A)	SIGNED	2	RPENUMDP	NUMBER OF CATEGORIES IN LIST
28	(1C)	UNSIGNED	2	RPEDPTOF	OFFSET TO CATEGORY LIST
30	(1E)	BITSTRING	1	RPELDAYS	DAYS TERMINAL MAY NOT BE USED (BIT 0 - SUNDAY, BIT 1 - MONDAY, ...)
31	(1F)	UNSIGNED	1	RPESCLVL	RESOURCE SECURITY LEVEL
32	(20)	CHARACTER	3	RPELOGNT	EARLIEST TIME TERMINAL MAY BE USED (HHMM)
35	(23)	CHARACTER	3	RPELOGFT	LATEST TIME TERMINAL MAY BE USED (HHMM)
38	(26)	CHARACTER	8	RPENTFY	USERID TO NOTIFY WHEN THIS PROFILE DENIES ACCESS
46	(2E)	CHARACTER	3	RPETZONE	TIME OFFSET OF TERMINAL FROM CPU. + = EAST - = WEST.
49	(31)	BITSTRING	1	RPEFLAGS	FLAGS FOR IN STORE PROFILE

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		`1...  ....`		RPEFWARN	WARN OPTION SPECIFIED?
		`.111  1111`		*	RESERVED
50	(32)	CHARACTER	8	RPESCLBL	SECLABEL
58	(3A)	UNSIGNED	2	RPESESOF	SESSION SEG DATA OFF
60	(3C)	UNSIGNED	2	RPESESLN	SESSION SEG DATA LEN
62	(3E)	UNSIGNED	2	RPEAC2NO	NUMBER OF OCCURRENCES
64	(40)	UNSIGNED	2	RPEAC2LN	CONDITIONAL ACCESS LIST LENGTH
66	(42)	UNSIGNED	2	RPEAC2OF	SECOND ACCESS LIST OFFSET
68	(44)	UNSIGNED	2	RPEMEMCT	NUMBER OF MEMBERS
70	(46)	UNSIGNED	2	RPEMEMLN	LENGTH OF MEMBER LIST
72	(48)	UNSIGNED	2	RPEMEMOF	OFFSET TO MEMBER LIST
74	(4A)	SIGNED	2	RPESE2LN	MORE SESSION DATA LENGTH
76	(4C)	SIGNED	2	RPESE2OF	MORE SESSION DATA OFFSET
78	(4E)	CHARACTER		RPEEND	END OF FIXED PART OF ELEMENT

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	RPEINST	INSTALLATION DATA VARIABLE LENGTH PORTION
0	(0)	UNSIGNED	1	RPEINSTL	INSTALLATION DATA LENGTH
1	(1)	CHARACTER	*	RPEINSTD	INSTALLATION DATA STRING

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	RPEAPPL	APPLICATION DATA VARIABLE LENGTH PORTION
0	(0)	UNSIGNED	1	RPEAPPLL	APPLICATION DATA LENGTH
1	(1)	CHARACTER	*	RPEAPPLD	APPLICATION DATA STRING

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	9	RPEACCLE (*)	ACCESS LIST
0	(0)	CHARACTER	8	RPEAUSR	USER/GROUP ID
8	(8)	BITSTRING	1	RPEACS	ACCESS AUTHORITY

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	2	RPEDPTD (*)	CATEGORY LIST
0	(0)	SIGNED	2	RPEDEPT	CATEGORY

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	RPESESSN	SESSION DATA
0	(0)	CHARACTER	10	RPESEFIX	FIXED LEN SESSION FIELDS
0	(0)	CHARACTER	4	RPEKYDAT	DATE KEY WAS LAST CHANGED
4	(4)	SIGNED	2	RPEKYINT	# DAYS UNTIL KEY EXPIRES
6	(6)	SIGNED	2	RPEMFAIL	MAX # OF FAILED ATTEMPTS
8	(8)	BITSTRING	1	RPESLSFG	SESSION FLAGS
9	(9)	UNSIGNED	1	RPESKYLN	LENGTH OF SESSION KEY
10	(A)	CHARACTER	*	RPESEVAR	VARIABLE LEN FIELDS
10	(A)	CHARACTER	*	RPESNKEY	SESSION KEY

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	RPEACL2	SECOND ACCESS LIST
0	(0)	CHARACTER	20	RPEA2FIX	FIXED LENGTH PORTION OF SECOND ACCESS LIST
0	(0)	CHARACTER	8	RPEA2PGM	PROGRAM NAME OR FLAGS
0	(0)	CHARACTER	1	RPEPGFLG	FLAG BYTE
1	(1)	CHARACTER	7	RPEA2RST	THE REST OF NAME OR FLAGS
8	(8)	CHARACTER	8	RPEA2USR	USERID
16	(10)	BITSTRING	1	RPEA2ACA	ACCESS AUTHORITY
17	(11)	UNSIGNED	2	RPEA2CNT	ACCESS COUNT FIELD

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
19	(13)	UNSIGNED	1	RPEA2VRL	VARIABLE AREA LENGTH
20	(14)	CHARACTER	*	RPEA2VAR	VARIABLE AREA
20	(14)	CHARACTER	8	RPEA2CLI	CLASS ID.
28	(1C)	CHARACTER	2	RPEA2RSV	RESERVED.
30	(1E)	UNSIGNED	1	RPEA2ELN	ENTITY LENGTH
31	(1F)	CHARACTER	*	RPEA2ENT	ENTITY

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	RPEMEM	MEMBER LIST
0	(0)	UNSIGNED	1	RPEMEML	MEMBER LENGTH
1	(1)	CHARACTER	*	RPEMEMBR	MEMBER

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	1	RPESESS2	MORE SESSION
0	(0)	CHARACTER	1	RPESE2FX	MORE SESSION FIXED FIELDS
0	(0)	BITSTRING	1	RPESCONV	CONVERSATION SECURITY

Constants

Len	Type	Value	Name	Description
1	DECIMAL	0	RPEA2DAT	FLAG DATA EQUATE

Cross Reference

Name	Hex Offset	Hex Value	Level
RACRNE	0		1
RACRPE	0		1
RACRSE	0		1
RACRTE	0		1
RNEBAL	C		2
RNEKEY	10		2
RNELEFT	0		2

Name	Hex Offset	Hex Value	Level
RNEPROF	4		2
RNERIGHT	8		2
RNEUP	C		3
RPEACCLE	0		1
RPEACCNO	A		2
RPEACCOF	C		2
RPEACL2	0		1
RPEACS	8		2
RPEAC2LN	40		2
RPEAC2NO	3E		2
RPEAC2OF	42		2
RPEAPPL	0		1
RPEAPPLD	1		2
RPEAPPLL	0		2
RPEAPPOF	10		2
RPEATTR	6		2
RPEAUDIT	7		3
RPEAUSR	0		2
RPEA2ACA	10		3
RPEA2CLI	14		3
RPEA2CNT	11		3
RPEA2ELN	1E		3
RPEA2ENT	1F		3
RPEA2FIX	0		2
RPEA2PGM	0		3
RPEA2RST	1		4
RPEA2RSV	1C		3
RPEA2USR	8		3
RPEA2VAR	14		2
RPEA2VRL	13		3
RPEDEPT	0		2
RPEDPTD	0		1
RPEDPTOF	1C		2
RPEEND	4E		2
RPEFLAGS	31		2
RPEFWARN	31	80	3

Name	Hex Offset	Hex Value	Level
RPEGAUD	8		3
RPEINSOF	E		2
RPEINST	0		1
RPEINSTD	1		2
RPEINSTL	0		2
RPEKYDAT	0		3
RPEKYINT	4		3
RPELDAYS	1E		2
RPELEVEL	9		3
RPELLEN	2		2
RPELOGFT	23		2
RPELOGNT	20		2
RPEMEM	0		1
RPEMEMBR	1		2
RPEMEMCT	44		2
RPEMEML	0		2
RPEMEMLN	46		2
RPEMEMOF	48		2
RPEMFAIL	6		3
RPENTFY	26		2
RPENUMDP	1A		2
RPEOWNER	12		2
RPEPGFLG	0		4
RPEPLEN	0		2
RPESCLBL	32		2
RPESCLVL	1F		2
RPESCONV	0		3
RPESEFIX	0		2
RPESESLN	3C		2
RPESESOF	3A		2
RPESESSN	0		1
RPESESS2	0		1
RPESEVAR	A		2
RPESE2FX	0		2
RPESE2LN	4A		2
RPESE2OF	4C		2

Name	Hex Offset	Hex Value	Level
RPESKYLN	9		3
RPESLSFG	8		3
RPESNKEY	A		3
RPETZONE	2E		2
RPEUACC	6		3
RPEUCNT	4		2
RSENEXT	0		2
RSEPOOL	6		2
RSESIZE	4		2
RSESTORE	8		2
RTEASPN	12		2
RTECLASS	4		2
RTEGENL	14		2
RTEGNUM	1C		2
RTENEXT	0		2
RTENSPN	11		3
RTEPSPN	10		3
RTESIZE	18		2
RTESPNS	10		2
RTESTORE	C		2
RTETREE	8		2

RCVT

NOT programming interface information

RCVT except for the following fields, which are general-use programming interface information

- RCVT
- RCVTAPTR
- RCVTDATP
- RCVTFLGS
- RCVTFLG1
- RCVTFRCP
- RCVTID
- RCVTISTL
- RCVTJALL
- RCVTJCHK
- RCVTJSYS
- RCVTJUND
- RCVTJXAL

- RCVTMFLG
- RCVTPNL0
- RCVTRELS
- RCVTREXP
- RCVTRNA
- RCVTROFF
- RCVTSTAT
- RCVTSTA1
- RCVTTAPE
- RCVTTDSN
- RCVTVERS
- RCVTVRMF
- RCVTVRN
- RCVTVRMN
- RCVTWUID

Note:

1. To Application Programmers: The RCVT fields listed above are general-use programming interfaces for input only, with the following exceptions:

   RCVTISTL and RCVTAPTR can be both input and output
   RCVTREXP and RCVTFRCP are not part of the application programming interface.
2. To External Security Managers (ESM) (such as RACF or an ESM that is functionally compatible with RACF): The RCVT fields listed above are general-use programming interfaces for both input and output. The ESM is responsible for creating the RCVT, attaching it to the communication vector table (CVT), and putting appropriate data into these fields in order to be compatible with RACF and the way that IBM products use the RCVT.

End of NOT programming interface information

Common Name:	RACF Communication Vector Table
Macro ID:	ICHPRCVT
DSECT Name:	RCVT
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	RCVT Offset: 0 Length: 4
Subpool and Key:	SQA and key 0
Size:	768 bytes
Created by:	RACF initialization or equivalent
Pointed to by:	CVTRAC
Serialization:	None
Function:	Communication area for information global to RACF functions (or equivalent product functions).

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	768	RCVT	LOCATED THROUGH CVT
0	(0)	CHARACTER	4	RCVTID	EBCDIC ID
4	(4)	ADDRESS	4	RCVTDCB	PTR DCB OF RACF DATA SET
8	(8)	ADDRESS	4	RCVTDEB	PTR DEB OF RACF DATA SET
12	(C)	ADDRESS	4	RCVTINDX	PTR RACF RESIDENT INDEX TABLE OR ZERO IF NO INDEX BLOCKS RESIDENT
16	(10)	ADDRESS	4	RCVTTEMP	PTR RACF INCORE TEMPLATE TABLE
20	(14)	ADDRESS	4	RCVTHDR	PTR RACF INCORE DS HEADER RECORD OR ZERO IF RACF DATA SET IS ON A SHARED DEVICE
24	(18)	ADDRESS	4	RCVTRIX	PTR RACINIT INSTALL. EXIT RTN
28	(1C)	ADDRESS	4	RCVTRCX	PTR RACCHK INSTALL. EXIT RTN
32	(20)	ADDRESS	4	RCVTRDX	PTR RACDEF INSTALL. EXIT RTN
36	(24)	ADDRESS	4	RCVTRUCB	PTR UCB OF RACF DATA SET
40	(28)	SIGNED	4	RCVTXLEN	LENGTH OF INCORE INDEX RELATED CONTROL BLOCKS
44	(2C)	ADDRESS	4	RCVTBAM	LOCATES INCORE BAM INFORMATION
48	(30)	ADDRESS	4	RCVTISTL	RESERVED FOR INSTALLATION
52	(34)	ADDRESS	1	RCVTDSNL	LENGTH OF RAC DATA SET NAME
53	(35)	BITSTRING	1	RCVTSTAT	STATUS
		`1...  ....`		RCVTRNA	RACF NOT ACTIVE
		`.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

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.... ..1.		RCVTNADS	NO ADSP PROTECTION
		1		RCVTEGN	EGN SUPPORT IN EFFECT
54	(36)	SIGNED	2	RCVTNREC	# RECORDS PER TRACK -RACF DS
56	(38)	CHARACTER	44	RCVTDSN	DSN OF RACF DATA SET
100	(64)	CHARACTER	44	RCVTUADS	DSN OF UADS DATA SET OR ZERO
144	(90)	CHARACTER	6	RCVTUVOL	VOLID OF UADS DATA SET OR ZERO
150	(96)	BITSTRING	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)	BITSTRING	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

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		.... .1..		RCVTATAP	AUDIT TAPEVOL CLASS
		.... ..1.		RCVTATER	AUDIT TERMINAL CLASS
		1		RCVTAOPR	AUDIT OPERATIONS ATTRIBUTE
152	(98)	BITSTRING	1	RCVTAXTA	RESERVED
153	(99)	BITSTRING	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
		.... ..1.		RCVTWUID	RACF WORK UNIT IDENTITY SUPPORT EXISTS.
		1		*	RESERVED.
154	(9A)	BITSTRING	1	RCVTEROP	RACF TERMINAL OPTIONS
		1...		RCVTTERP	TERMINAL AUTHORIZATION CHECKING
		.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

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
155	(9B)	ADDRESS	1	RCVTPINV	GLOBAL MAX PASSWORD INTERVAL VALUE - VALID RANGE 1-254
156	(9C)	ADDRESS	4	RCVTRAU0	PTR TO AUDITING MODULE
160	(A0)	ADDRESS	4	RCVTRIXP	PTR TO RACINIT POST PROCESSING INSTALLATION EXIT
164	(A4)	ADDRESS	4	RCVTRCXP	PTR TO RACCHK POST PROCESSING INSTALLATION EXIT
168	(A8)	ADDRESS	4	RCVTRID0	PTR TO MSC VERIFY RTN
172	(AC)	ADDRESS	1	RCVTVERS	VERSION INDICATOR: HIGH NIBBLE IS THE VERSION NUMBER, (0=VERSION 1), AND THE LOW NIBBLE IS THE RELEASE NUMBER 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 8 - VERSION 1 RELEASE 8
		1111		RCVTVRN	VERSION NUMBER IN HIGH NIBBLE
		 1111		RCVTRELS	RELEASE NUMBER IN LOW NIBBLE
173	(AD)	CHARACTER	3	RCVTEXTA	RESERVED
176	(B0)	ADDRESS	4	RCVTAPTR	ADDRESS FIELD RESERVED FOR APPLICATION USE
180	(B4)	ADDRESS	4	RCVTNCX	PTR NAMING CONVENTION EXIT
184	(B8)	ADDRESS	4	RCVTNCDX	PTR NAMING CONVENTION EXIT FOR DELETE FUNCTION
188	(BC)	ADDRESS	4	RCVTCDTP	PTR TO CLASS DESC TABLE
192	(C0)	ADDRESS	4	RCVTREXP	PTR TO RACSTAT MODULE
196	(C4)	ADDRESS	4	RCVTFRCP	PTR TO FRACHECK MODULE
200	(C8)	ADDRESS	4	RCVTFRXP	PTR TO FRACHECK EXIT
204	(CC)	ADDRESS	4	RCVTRLX	PTR TO RACLIST PRE-EXIT
208	(D0)	ADDRESS	4	RCVTRLXP	PTR TO RACLIST SELECTION EXIT
212	(D4)	BITSTRING	4	RCVTCSTA	CLASS STATISTICS OPTION

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
216	(D8)	BITSTRING	4	RCVTCAUD	CLASS AUDITING OPTIONS
220	(DC)	BITSTRING	4	RCVTCPRO	CLASS PROTECTION OPTION
224	(E0)	ADDRESS	4	RCVTDSDT	PTR TO DATA SET DESCRIPTOR TABLE
228	(E4)	ADDRESS	4	RCVTRNGP	PTR TO RANGE TABLE
232	(E8)	ADDRESS	4	RCVTAUTP	PTR TO RACF AUTHORIZED CALLER TABLE ICHAUTAB
236	(EC)	ADDRESS	4	RCVTPWDX	PTR TO RACF PASSWORD EXIT.
240	(F0)	UNSIGNED	1	RCVTHIST	NUMBER OF PASSWORD GENERATIONS TO MAINTAIN AND CHECK AGAINST.
241	(F1)	UNSIGNED	1	RCVTRVOK	NUMBER OF CONSECUTIVE UNSUCCESSFUL ATTEMPTS BEFORE REVOKING A USERID.
242	(F2)	UNSIGNED	1	RCVTWARN	PASSWORD WARNING VALUE.
243	(F3)	UNSIGNED	1	RCVTINAC	INACTIVATE INTERVAL.
244	(F4)	CHARACTER	10	RCVTSNTX (8)	PASSWORD SYNTAX RULES.
244	(F4)	UNSIGNED	1	RCVTSLEN	STARTING LENGTH VALUE.
245	(F5)	UNSIGNED	1	RCVTELEN	ENDING LENGTH VALUE.
246	(F6)	CHARACTER	8	RCVTRULS	CONTENT RULES.
246	(F6)	CHARACTER	1	RCVTRUL1	CONTENT RULE.
247	(F7)	CHARACTER	1	RCVTRUL2	CONTENT RULE.
248	(F8)	CHARACTER	1	RCVTRUL3	CONTENT RULE.
249	(F9)	CHARACTER	1	RCVTRUL4	CONTENT RULE.
250	(FA)	CHARACTER	1	RCVTRUL5	CONTENT RULE.
251	(FB)	CHARACTER	1	RCVTRUL6	CONTENT RULE.
252	(FC)	CHARACTER	1	RCVTRUL7	CONTENT RULE.
253	(FD)	CHARACTER	1	RCVTRUL8	CONTENT RULE.
324	(144)	CHARACTER	4	RCVTMDEL	MODEL OPTIONS.
324	(144)	BITSTRING	1	*	OPTIONS.
		`1...  ....`		RCVTMGDG	MODEL-GDG IN EFFECT.
		`.1..  ....`		RCVTMUSR	MODEL-USER IN EFFECT.
		`..1.  ....`		RCVTMGRP	MODEL-GROUP IN EFFECT.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		...1 1111		*	RESERVED.
325	(145)	BITSTRING	1	*	RESERVED.
326	(146)	BITSTRING	1	*	RESERVED.
327	(147)	BITSTRING	1	*	RESERVED.
328	(148)	BITSTRING	1	RCVTWCNT	NUMBER OF VSL ENTRIES
329	(149)	BITSTRING	1	RCVTOPTX	OPTIONS.
		1...		RCVTLGRP	LIST-OF-GRPS CHKING ACTIVE.
		.111 1111		*	RESERVED
330	(14A)	CHARACTER	2	* (2)	RESERVED.
332	(14C)	ADDRESS	4	RCVTDATP	PTR TO 4 BYTE DATE CONVERSION ROUTINE
336	(150)	CHARACTER	8	RCVTVSL (4)	VSL ENTRIES
368	(170)	SIGNED	4	RCVTCGSN	NUMBER OF CONNECT-REMOVE COMMANDS ISSUED THAT ALTERED A USER'S AUTHORITY.
372	(174)	BITSTRING	4	RCVTCGEN	CLASS MASK FOR GENERIC PROFILE CHECKING
376	(178)	BITSTRING	4	RCVTCGCM	CLASS MASK FOR GENERIC COMMAND PROCESSING
380	(17C)	ADDRESS	4	RCVTRDXP	PTR TO RACDEF POST PROCESSING INSTALLATION EXIT- ICHRDX02
384	(180)	ADDRESS	4	RCVTFPB	BASE FOR FASTPATH TABLE.
388	(184)	BITSTRING	4	RCVTFPTH	CLASS FASTPATH OPTIONS.
392	(188)	BITSTRING	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

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		.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
394	(18A)	BITSTRING	1	*	RESERVED.
396	(18C)	UNSIGNED	2	RCVTRTPD	SYSTEM SECURITY RETENTION PERIOD
398	(18E)	UNSIGNED	1	RCVTSLVL	SECURITY LEVEL FOR ERASE-ON- SCRATCH
399	(18F)	UNSIGNED	1	RCVTQLLN	LENGTH OF SINGLE LEVEL DATASET NAME PREFIX
400	(190)	CHARACTER	9	RCVTQUAL	INSTALLATION CONTROLLED PREFIX FOR SINGLE LEVEL DATASET NAMES, PLUS PERIOD FOR LEVEL
409	(199)	UNSIGNED	1	RCVTSLAU	SECLEVEL TO AUDIT
410	(19A)	BITSTRING	1	RCVTMFLG	MISCELLANEOUS FLAGS
		1...		RCVTVRMF	RACF VERSION, RELEASE, AND MODIFICATION FLAG FOR THE ICQ (TSO) SUPPORT IN 1.8.1
		.1..		RCVT310U	RUNNING MVS/SP 3.1.0 OR UP
		.... 1...		RCVTD4OK	DATE CONVERSION ROUTINE IS AVAILABLE

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		.... ..1.		RCVT4INF	SUPPORT FOR 4 BYTE DATES ON PROGRAMMING INTERFACES IS AVAILABLE
411	(19B)	CHARACTER	1	*	RESERVED.
412	(19C)	ADDRESS	4	RCVTSPT	POINTER TO THE STARTED PROCEDURES TABLE (ICHRIN03)
416	(1A0)	ADDRESS	4	RCVTDESX	POINTER TO THE PASSWORD ENCRYPTION INSTALLATION EXIT (ICHDEX01)
420	(1A4)	ADDRESS	4	RCVTNTAB	POINTER TO THE NAMING CONVENTION TABLE (ICHNCV00)
424	(1A8)	ADDRESS	4	RCVTNRTN	POINTER TO THE NAMING CONVENTION ROUTINE (ICHNRT00)
428	(1AC)	ADDRESS	4	RCVTFRX2	ADDRESS OF THE FRACHECK POST- PROCESSING INSTALLATION EXIT (ICHRFX02)
432	(1B0)	CHARACTER	8	RCVTPROB	ADDRESSES OF CONTROLLED PROGRAMS LIST ANCHOR BLOCKS
432	(1B0)	ADDRESS	4	RCVTCISP	ADDRESS OF CURRENT ANCHOR FOR CONTROLLED PROGRAMS LIST
436	(1B4)	ADDRESS	4	RCVTOISP	ADDRESS OF OLD ANCHOR FOR CONTROLLED PROGRAMS LIST
440	(1B8)	CHARACTER	8	RCVTSWPW	PASSWORD FOR RVARY SWITCH
448	(1C0)	CHARACTER	8	RCVTINPW	PASSWORD FOR RVARY STATUS
456	(1C8)	ADDRESS	4	RCVTLARP	PTR TO LINKAGE ASSIST ROUTINE FOR INSTAL EXITS (ICHLAR00)
460	(1CC)	ADDRESS	4	RCVTCTV0	ADDRESS OF TVTOC UTILITY (ICHCTV00)
464	(1D0)	ADDRESS	4	RCVTPNL0	POINTER TO PROFILE NAME LIST ROUTINE
468	(1D4)	BITSTRING	16	RCVTLRCL	CDT-ANCHORED RACLISTED PROFILE CLASS MASK, ON IF ACTIVE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
484	(1E4)	BITSTRING	16	RCVTLGNL	CDT-ANCHORED GENLISTED PROFILE CLASS MASK, ON IF ACTIVE
500	(1F4)	BITSTRING	16	RCVTLCST	CLASS STATISTICS OPTION MASK LONG VERSION, ON IF ACTIVE
516	(204)	BITSTRING	16	RCVTLCAU	CLASS AUDITING OPTION MASK LONG VERSION, ON IF ACTIVE
532	(214)	BITSTRING	16	RCVTLCPR	CLASS PROTECTION OPTION MASK LONG VERSION, ON IF ACTIVE
548	(224)	BITSTRING	16	RCVTLCGE	CLASS MASK FOR GENERIC PROFILE CHECKING LONG VERSION, ON IF ACTIVE
564	(234)	BITSTRING	16	RCVTLCGC	CLASS MASK FOR GENERIC COMMAND CHECKING LONG VERSION, ON IF ACTIVE
580	(244)	BITSTRING	16	RCVTLFPT	CLASS FASTPATH OPTION MASK LONG VERSION, ON IF ACTIVE
596	(254)	ADDRESS	4	RCVTGLS1	ADDRESS OF GENLIST DELETE ROUTINE (ICHGLS01)
600	(258)	ADDRESS	4	RCVTRCVX	ADDRESS OF RCVT EXTENSION AREA
604	(25C)	ADDRESS	4	RCVTLAR2	ADDRESS OF ICHLAR02
608	(260)	ADDRESS	4	RCVTFLT0	ADDRESS OF IRRFLT00
612	(264)	ADDRESS	4	RCVTFLT1	ADDRESS OF IRRFLT01
616	(268)	CHARACTER	4	RCVTVRMN	RACF VERSION, RELEASE, AND MODIFICATION NUMBER (VRRM)
620	(26C)	SIGNED	4	RCVTVMSP	ICB SYNC COUNT VM 370
624	(270)	SIGNED	4	RCVTVMXA	ICB SYNC COUNT VM XA
628	(274)	BITSTRING	1	RCVTFLG2	RACF 1.9.0 SETROPTS OPTIONS
		1...		RCVTSLCL	SETROPTS SECLABELCONTROL - ON if active
		.1..		RCVTCATD	SETROPTS CATDSNS - ON if active
		..1.		RCVTMLQT	SETROPTS MLQUIET - ON if active

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		...1		RCVTMLST	SETROPTS MLSTABLE - ON if active
		.... 1...		RCVTMLS	SETROPTS MLS - ON if active
		.... .1..		RCVTMLAC	SETROPTS MLACTIVE - ON if active
		.... ..1.		RCVTGNOW	SETROPTS GENERICOWNER - ON if active
		1		RCVTAUSL	SETROPTS SECLABELAUDIT - ON if active
629	(275)	BITSTRING	1	RCVTLOGD	LOGOPTIONS FOR DATASET
		1...		RCVTDLGA	LOGOPTIONS "ALWAYS" FOR THE DATASET CLASS
		.1..		RCVTDLGN	LOGOPTIONS "NEVER" FOR THE DATASET CLASS
		..1.		RCVTDLGS	LOGOPTIONS "SUCCESSES" FOR THE DATASET CLASS
		...1		RCVTDLGF	LOGOPTIONS "FAILURES" FOR THE DATASET CLASS
		 1111		*	RESERVED
630	(276)	SIGNED	2	RCVTSINT	LU Session Interval
632	(278)	BITSTRING	16	RCVTLGAL	SETROPTS "LOGOPTIONS ALWAYS" Class Mask, ON if active
648	(288)	BITSTRING	16	RCVTLNVR	SETROPTS "LOGOPTIONS NEVER" Class Mask, ON if active
664	(298)	BITSTRING	16	RCVTLGSU	SETROPTS "LOGOPTIONS SUCCESSES" Class Mask, ON if active
680	(2A8)	BITSTRING	16	RCVTLGFL	SETROPTS "LOGOPTIONS FAILURES" Class Mask, ON if active
696	(2B8)	CHARACTER	8	RCVTJSYS	USER-ID from the SETROPTS command JES(NJEUSERID (user-id))
704	(2C0)	CHARACTER	8	RCVTJUND	USER-ID from the SETROPTS command JES(UNDEFINEDUSER (user-id))
712	(2C8)	ADDRESS	4	RCVTTMP2	ADDRESS OF RDS TEMPLATES
716	(2CC)	ADDRESS	4	RCVTRCK4	ADDRESS OF IRRRCK04

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
720	(2D0)	ADDRESS	4	RCVTSVC0	ADDRESS OF ICHSVC00
724	(2D4)	ADDRESS	4	*	RESERVED
728	(2D8)	ADDRESS	4	*	RESERVED
732	(2DC)	ADDRESS	4	RCVTDX11	ADDRESS OF ICHDEX11
736	(2E0)	ADDRESS	4	RCVTXLT0	ADDRESS OF IRRRXT02
740	(2E4)	ADDRESS	4	RCVTGLS6	ADDRESS OF ICHGLS06
744	(2E8)	ADDRESS	4	RCVTDPTB	ADDRESS OF DYNAMIC PARSE TABLE
748	(2EC)	ADDRESS	4	RCVTRCK2	ADDRESS OF IRRRCK02
752	(2F0)	ADDRESS	4	RCVTRX10	Address of IRRRXT10
756	(2F4)	ADDRESS	4	RCVTRX11	Address of IRRRXT11
760	(2F8)	ADDRESS	4	RCVTDSPC	Address of IRRDSP00
764	(2FC)	BITSTRING	1	RCVTFL2X	Extension of RACF 1.9.0 SETROPTS options
		1...		RCVTCMPM	SETROPTS COMPATMODE - ON if active
		.1..		RCVTMLSF	SETROPTS MLS FAILURES/ WARNING - FAILURES if "ON" - (1) - WARNING if "OFF" - (0)
		..1.		RCVTMLAF	SETROP MLACTIVE FAILURES/ WARNING - FAILURES if "ON" - (1) - WARNING if "OFF" - (0)
		...1		RCVTCATF	SETROPT CATDSNS FAILURES/WARNING - FAILURES if "ON" - (1) - WARNING if "OFF" - (0)
		 1111		*	Reserved
765	(2FD)	BITSTRING	1	RCVTNJEF	NJE Flags
		1...		RCVTJWTO	Flag indicating WTO has been issued for NJE, if "ON" - (1)
		.111 1111		*	Reserved
766	(2FE)	CHARACTER	2	*	Reserved
768	(300)	CHARACTER		*	END OF RCVT

Constants

Len	Type	Value	Name	Description
1	DECIMAL	8	RCVTVERN	VERSION NUMBER VALUE: HIGH NIBBLE IS THE VERSION NUMBER, (0=VERSION 1), AND THE LOW NIBBLE IS THE RELEASE NUMBER
4	CHARACTER	5030	RCVTVRMC	RACF VERSION, RELEASE, AND MODIFICATION NUMBER
4	CHARACTER	1081	RCVTVR81	indicates RACF 1.8.1 (z/OS only)
4	CHARACTER	1090	RCVTVR19	indicates RACF 1.9.0
4	CHARACTER	1092	RCVTVR92	indicates RACF 1.9.2
4	CHARACTER	2010	RCVTVR21	indicates RACF 2.1.0 (z/OS only)
4	CHARACTER	2020	RCVTVR22	indicates RACF 2.2.0 (z/OS only)
4	CHARACTER	1100	RCVTV110	indicates RACF 1.10.0 (z/VM only)
4	CHARACTER	5030	RCVTV530	indicates RACF 5.30 (z/VM only)
4	CHARACTER	RCVT	RCVTIDC	EBCDIC RCVT ID, FOR THE RCVT CONTROL BLOCK

Cross Reference

Name	Hex Offset	Hex Value	Level
RCVT	0		1
RCVTADAS	97	08	3
RCVTADAT	97	10	3
RCVTAGRO	97	40	3
RCVTAOPR	97	01	3
RCVTAPTR	B0		2
RCVTATAP	97	04	3
RCVTATER	97	02	3
RCVTAUOP	97		2
RCVTAUSE	97	20	3
RCVTAUSL	274	01	3
RCVTAUTP	E8		2
RCVTAVIO	9A	20	3
RCVTAXTA	98		2
RCVTBAM	2C		2
RCVTCATD	274	40	3
RCVTCATF	2FC	10	3
RCVTCAUD	D8		2

Name	Hex Offset	Hex Value	Level
RCVTCDTP	BC		2
RCVTCGCM	178		2
RCVTCGEN	174		2
RCVTCGSN	170		2
RCVTCISP	1B0		3
RCVTCMPM	2FC	80	3
RCVTCPRO	DC		2
RCVTCSTA	D4		2
RCVTCTV0	1CC		2
RCVTDASD	96	40	3
RCVTDATP	14C	''2	
RCVTDCB	4		2
RCVTDEB	8		2
RCVTDESX	1A0		2
RCVTDGCM	96	10	3
RCVTDGEN	96	20	3
RCVTDLGA	275	80	3
RCVTDLGF	275	10	3
RCVTDLGN	275	40	3
RCVTDLGS	275	20	3
RCVTDPTB	2E8		2
RCVTDSDT	E0		2
RCVTDSN	38		2
RCVTDSNL	34		2
RCVTDSPC	2F8		2
RCVTDX11	2DC		2
RCVTD4OK	19A	08	3
RCVTEGN	35	01	3
RCVTELEN	F5		3
RCVTEOS	189	20	3
RCVTEOSA	189	08	3
RCVTEOSL	189	10	3
RCVTEROP	9A		2
RCVTEXTA	AD		2
RCVTFLGS	99		2
RCVTFLG1	188		2

Name	Hex Offset	Hex Value	Level
RCVTFLG2	274		2
RCVTFLT0	260		2
RCVTFLT1	264		2
RCVTFL2X	2FC		2
RCVTFPB	180		2
RCVTFPDS	188	80	3
RCVTFPTH	184		2
RCVTFRCP	C4		2
RCVTFRXP	C8		2
RCVTFRX2	1AC		2
RCVTGLS1	254		2
RCVTGLS6	2E4		2
RCVTGNOW	274	02	3
RCVTHDR	14		2
RCVTHIST	F0		2
RCVTID	0		2
RCVTINAC	F3		2
RCVTINDX	C		2
RCVTINPW	1C0		2
RCVTISTL	30		2
RCVTJALL	96	01	3
RCVTJCHK	96	02	3
RCVTJSYS	2B8		2
RCVTJUND	2C0		2
RCVTJWTO	2FD	80	3
RCVTJXAL	96	04	3
RCVTLARP	1C8		2
RCVTLAR2	25C		2
RCVTLCAU	204		2
RCVTLCGC	234		2
RCVTLCGE	224		2
RCVTLCPR	214		2
RCVTLCST	1F4		2
RCVTLFPT	244		2
RCVTLGAL	278		2
RCVTLGFL	2A8		2

Name	Hex Offset	Hex Value	Level
RCVTLGNL	1E4		2
RCVTLGRP	149	80	3
RCVTLGSU	298		2
RCVTLNVR	288		2
RCVTLOGD	275		2
RCVTLRCL	1D4		2
RCVTMDEL	144		2
RCVTMFLG	19A		2
RCVTMGDG	144	80	4
RCVTMGRP	144	20	4
RCVTMLAC	274	04	3
RCVTMLAF	2FC	20	3
RCVTMLQT	274	20	3
RCVTMLS	274	08	3
RCVTMLSF	2FC	40	3
RCVTMLST	274	10	3
RCVTMUSR	144	40	4
RCVTNADS	35	02	3
RCVTNCDX	B8		2
RCVTNCX	B4		2
RCVTNDSS	35	20	3
RCVTNDUP	99	10	3
RCVTNDVS	35	08	3
RCVTNJEF	2FD		2
RCVTNLS	35	40	3
RCVTNREC	36		2
RCVTNRTN	1A8		2
RCVTNTAB	1A4		2
RCVTNTMS	35	04	3
RCVTNTVS	35	10	3
RCVTOISP	1B4		3
RCVTOPTX	149		2
RCVTPINV	9B		2
RCVTPNL0	1D0		2
RCVTPRO	189	80	3
RCVTPROB	1B0		2

Name	Hex Offset	Hex Value	Level
RCVTPROF	189	40	3
RCVTPROG	18A	80	3
RCVTPWDX	EC		2
RCVTQLLN	18F		2
RCVTQUAL	190		2
RCVTRAU0	9C		2
RCVTRCK2	2EC		2
RCVTRCK4	2CC		2
RCVTRCVX	258		2
RCVTRCX	1C		2
RCVTRCXP	A4		2
RCVTRDHD	99	40	3
RCVTRDSN	96	08	3
RCVTRDX	20		2
RCVTRDXP	17C		2
RCVTRELS	AC	08	3
RCVTREXP	C0		2
RCVTRID0	A8		2
RCVTRIX	18		2
RCVTRIXP	A0		2
RCVTRLX	CC		2
RCVTRLXP	D0		2
RCVTRMSG	99	04	3
RCVTRNA	35	80	3
RCVTRNGP	E4		2
RCVTROFF	99	80	3
RCVTRTPD	18C		2
RCVTRUCB	24		2
RCVTRULS	F6		3
RCVTRUL1	F6		4
RCVTRUL2	F7		4
RCVTRUL3	F8		4
RCVTRUL4	F9		4
RCVTRUL5	FA		4
RCVTRUL6	FB		4
RCVTRUL7	FC		4

Name	Hex Offset	Hex Value	Level
RCVTRUL8	FD		4
RCVTRVOK	F1		2
RCVTRX10	2F0		2
RCVTRX11	2F4		2
RCVTSAUD	9A	10	3
RCVTSHR	99	20	3
RCVTSINT	276		2
RCVTSLAU	199		2
RCVTSLCL	274	80	3
RCVTSLEN	F4		3
RCVTSLVL	18E		2
RCVTSNTX	F4		2
RCVTSPT	19C		2
RCVTSTAT	35		2
RCVTSTA1	96		2
RCVTSVC0	2D0		2
RCVTSWPW	1B8		2
RCVTTAPE	96	80	3
RCVTTDSN	188	40	3
RCVTTEMP	10		2
RCVTTERP	9A	80	3
RCVTTMP2	2C8		2
RCVTTUAC	9A	40	3
RCVTUADS	64		2
RCVTUVOL	90		2
RCVTVERS	AC		2
RCVTVMSP	26C		2
RCVTVMXA	270		2
RCVTVRMF	19A	80	3
RCVTVRMN	268		2
RCVTVRN	AC	80	3
RCVTVSL	150		2
RCVTWARN	F2		2
RCVTWCNT	148		2
RCVTWUID	99	02	3
RCVTXLEN	28		2

Name	Hex Offset	Hex Value	Level
RCVTXLT0	2E0		2
RCVT24MD	99	08	3
RCVT310U	19A	40	3
RCVT4INF	19A	02	3

RDDFL

Common Name:	Request-specific portion of the RACROUTE REQUEST=DEFINE parameter list
Macro ID:	ICHRDDFL
DSECT Name:	RDDFLIST
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Varies depending on release parameter specified
Created by:	RACROUTE REQUEST=DEFINE macro
Pointed to by:	Address of SAFP plus the offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=DEFINE routine.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	48	RDDFLIST	RACDEF PARAMETER LIST
0	(0)	ADDRESS	4	RDDFINSW	ADDRESS OF INSTALLATION DATA
0	(0)	UNSIGNED	1	RDDFLENG	LENGTH OF PARAMETER LIST
1	(1)	ADDRESS	3	RDDFINST	INSTALLATION DATA
4	(4)	SIGNED	4	RDDFENTW	ENTITY NAME ADDRESS WORD
4	(4)	UNSIGNED	1	RDDFFLGS	FLAGS BYTE
		`11..  ....`		RDDFCHGV	TYPE=CHGVOL.
		`1...  ....`		RDDFTDEL	TYPE=DELETE
		`.1..  ....`		RDDFTADV	TYPE=ADDVOL

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		..1.		RDDFOLDV	OLDVOL SPECIFIED
		...1		RDDFNEWN	NEWNAME SPECIFIED
		.... 1...		RDDF31IN	31-BIT ADDRESS LIST INDICATOR
		.... .1..		RDDFDSTV	DSTYPE=V
		.... ..1.		RDDFMDEL	DSTYPE=M
		1		RDDFSPEC	SPECIAL=YES
5	(5)	ADDRESS	3	RDDFENT	ENTITY NAME ADDRESS
8	(8)	ADDRESS	4	RDDFOVOL	OLD VOLSER ADDR
8	(8)	ADDRESS	4	RDDFNNMX	NEWNAMX ADDRESS
8	(8)	ADDRESS	4	RDDFNNAM	NEWNAME ADDRESS
12	(C)	ADDRESS	4	RDDFVSER	NEW VOLSER ADDRESS
16	(10)	ADDRESS	4	RDDFCLNW	CLASS NAME ADDRESS
20	(14)	ADDRESS	4	RDDFMENX	MODEL ENTITYX ADDRESS
20	(14)	ADDRESS	4	RDDFMENT	MODEL ENTITY ADDRESS
24	(18)	ADDRESS	4	RDDFMVOL	MODEL VOLSER ADDRESS
28	(1C)	ADDRESS	4	RDDFACEE	ACEE ADDRESS
32	(20)	ADDRESS	4	RDDFUNIT	UNIT INFORMATION ADDRESS.
36	(24)	BITSTRING	1	RDDFUACC	UACC FLAGS.
		1...		RDDFALTR	ALTER AUTHORITY.
		.1..		RDDFCNTL	CONTROL AUTHORITY.
		..1.		RDDFUPD	UPDATE AUTHORITY.
		...1		RDDFREAD	READ AUTHORITY.
		.... 1...		RDDFEXEC	EXEC AUTHORITY. (TURNED ON TOGETHER WITH NONE)
		.... .11.		*	RESERVED.
		1		RDDFNONE	NONE AUTHORITY.
37	(25)	UNSIGNED	1	RDDFLVL	LEVEL VALUE. 00 TO 99.

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
38	(26)	BITSTRING	1	RDDFAUDT	AUDIT FLAGS.
		1...		RDDFALL	AUDIT ALL ACCESSES.
		.1..		RDDFSUCC	AUDIT SUCCESSFUL ACCESS.
		..1.		RDDFFAIL	AUDIT ACCESSES THAT FAIL.
		...1		RDDFANON	NO AUDITING.
		.... 11..		RDDFQS	SUCCESS QUALIFIER
		11		RDDFQF	FAILURE QUALIFIER
39	(27)	BITSTRING	1	RDDFFLG2	2ND FLAG BYTE
		1...		RDDFRFI	RACFIND PARAMETER GIVEN
		.1..		RDDFRFIY	RACFIND=YES
		..1.		RDDFCHKA	CHKAUTH=YES
		...1		RDDFTAPE	DSTYPE=T GIVEN
		.... 1...		RDDFEOS	ERASE=YES GIVEN
		.... .1..		RDDFMGEN	MGENER PARAMETER GIVEN B'0'=ASIS B'1'=YES
		.... ..1.		RDDFWARN	WARNING=YES GIVEN
		1		RDDFGEN	GENERIC=YES GIVEN
40	(28)	ADDRESS	4	RDDFOWNR	OWNER ADDRESS.
44	(2C)	ADDRESS	4	RDDFDATA	INSTALLATION DATA ADDRESS
48	(30)	CHARACTER		RDDFEND	END OF V1.4 LIST

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
48	(30)	STRUCTURE	8	RDDF31	31-BIT-ADDRESS SAF EXTENSION

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
48	(30)	ADDRESS	4	RDDFIN31	31-BIT INSTALLATION DATA ADDRESS
52	(34)	ADDRESS	4	RDDFENTX	31-BIT ENTITYX NAME ADDRESS
52	(34)	ADDRESS	4	RDDFEN31	31-BIT ENTITY NAME ADDRESS
56	(38)	CHARACTER		RDD31END	END OF 31 BIT LIST

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
8	(8)	STRUCTURE	48	RDDF17	RACF 1.7 PARAMETER LIST EXTENSION
8	(8)	ADDRESS	4	RDDFACC1	ADDR OF ACCLVL (1ST)
12	(C)	ADDRESS	4	RDDFACC2	ADDR OF ACCLVL (2ND)
16	(10)	ADDRESS	4	RDDFSLVL	ADDR OF SECLVL DATA
20	(14)	ADDRESS	4	RDDFCATG	ADDR OF CATEGORY DATA
24	(18)	ADDRESS	4	RDDFEXDT	ADDR OF EXPIR DATE
28	(1C)	SIGNED	2	RDDFFSEQ	FILESEQ VALUE
30	(1E)	BITSTRING	1	RDDFFLGT	TAPE FLAG BYTE
		11..		RDDFTLBL	TAPELBL SPECIFIED NL=B'01' STD=B'00' BLP=B'10'
		..11 11..		*	RESERVED
		.... ..1.		RDDFEXPX	EXTENDED EXPDT INDICATOR B'1'=EXTENDED EXPDT FORMAT (CCYYDDDF) B'0'=STANDARD EXPDT FORMAT (YYDDDF)
		1		RDDFEXP	EXPDT/RETPD VALUE B'1'=EXPDT B'0'=RETPD
31	(1F)	BITSTRING	1	RDDFISUR	RACDEF ISSUER FLAG BYTE
		1...		RDDFISCM	B'1'=RACF COMMAND ISSUED RACDEF
		.111 1111		*	RESERVED
32	(20)	ADDRESS	4	RDDFMCLS	ADDR OF MCLASS VALUE
36	(24)	ADDRESS	4	RDDFNOTF	ADDR OF NOTIFY ID
40	(28)	ADDRESS	4	RDDFSTCL	RESERVED

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
44	(2C)	ADDRESS	4	RDDFMGCL	RESERVED
48	(30)	ADDRESS	4	RDDFRSOW	RESERVED
52	(34)	BITSTRING	1	RDDFENV	RESERVED FLAGS
		`1... ....`		RDDFVRFY	RESERVED
		`.1.. ....`		RDDFIENX	ENTITYX SPECIFIED
		`..1. ....`		RDDFIMEX	MENTX SPECIFIED
		`...1 ....`		RDDFINMX	NEWNAMX SPECIFIED
53	(35)	UNSIGNED	1	* (3)	RESERVED
56	(38)	CHARACTER		RDD17END	END OF V1.7 LIST

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
48	(30)	STRUCTURE	20	RDDF18X	RACF 1.8X PARAMETER LIST EXTENSION
48	(30)	ADDRESS	4	RDDFDDPR	DDNAME POINTER
52	(34)	ADDRESS	4	RDDFSLAB	POINTER TO SECLABEL
56	(38)	CHARACTER	12	*	UNSUED
68	(44)	CHARACTER		RDD8XEND	END OF V1.8X

Cross Reference

Name	Hex Offset	Hex Value	Level
RDDFACC1	8		2
RDDFACC2	C		2
RDDFACEE	1C		2
RDDFALL	26	80	3
RDDFALTR	24	80	3
RDDFANON	26	10	3
RDDFAUDT	26		2
RDDFCATG	14		2
RDDFCHGV	4	80	4
RDDFCHKA	27	20	3

Name	Hex Offset	Hex Value	Level
RDDFCLNW	10		2
RDDFCNTL	24	40	3
RDDFDATA	2C		2
RDDFDDPR	30		2
RDDFDSTV	4	04	4
RDDFEND	30		2
RDDFENT	5		3
RDDFENTW	4		2
RDDFENTX	34		2
RDDFENV	34		2
RDDFEN31	34		3
RDDFEOS	27	08	3
RDDFEXDT	18		2
RDDFEXEC	24	08	3
RDDFEXP	1E	01	3
RDDFEXPX	1E	02	3
RDDFFAIL	26	20	3
RDDFFLGS	4		3
RDDFFLGT	1E		2
RDDFFLG2	27		2
RDDFFSEQ	1C		2
RDDFGEN	27	01	3
RDDFIENX	34	40	3
RDDFIMEX	34	20	3
RDDFINMX	34	10	3
RDDFINST	1		3
RDDFINSW	0		2
RDDFIN31	30		2
RDDFISCM	1F	80	3
RDDFISUR	1F		2
RDDFLENG	0		3
RDDFLIST	0		1
RDDFLVL	25		2
RDDFMCLS	20		2
RDDFMDEL	4	02	4
RDDFMENT	14		3

Name	Hex Offset	Hex Value	Level
RDDFMENX	14		2
RDDFMGCL	2C		2
RDDFMGEN	27	04	3
RDDFMVOL	18		2
RDDFNEWN	4	10	4
RDDFNNAM	8		4
RDDFNNMX	8		3
RDDFNONE	24	01	3
RDDFNOTF	24		2
RDDFOLDV	4	20	4
RDDFOVOL	8		2
RDDFOWNR	28		2
RDDFQF	26	02	3
RDDFQS	26	08	3
RDDFREAD	24	10	3
RDDFRFI	27	80	3
RDDFRFIY	27	40	3
RDDFRSOW	30		2
RDDFSLAB	34		2
RDDFSLVL	10		2
RDDFSPEC	4	01	4
RDDFSTCL	28		2
RDDFSUCC	26	40	3
RDDFTADV	4	40	5
RDDFTAPE	27	10	3
RDDFTDEL	4	80	5
RDDFTLBL	1E	80	3
RDDFUACC	24		2
RDDFUNIT	20		2
RDDFUPD	24	20	3
RDDFVRFY	34	80	3
RDDFVSER	C		2
RDDFWARN	27	02	3
RDDF17	8		1
RDDF18X	30		1
RDDF31	30		1

Name	Hex Offset	Hex Value	Level
RDDF31IN	4	08	4
RDD17END	38		2
RDD31END	38		2
RDD8XEND	44		2

RIPL

Common Name:	Request-specific portion of the RACROUTE REQUEST=VERIFY, VERIFYX, or TOKENBLD parameter list
Macro ID:	IRRPRIPL
DSECT Name:	INITPARM
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Variable depending on release and function
Created by:	RACROUTE REQUEST=VERIFY, VERIFYX, or TOKENBLD macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=VERIFY, VERIFYX, or TOKENBLD routine.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	28	INITPARM	
0	(0)	ADDRESS	1	INITLEN	PARM LIST LENGTH (28)
1	(1)	UNSIGNED	1	INITSUB#	SUBPOOL FOR ACEE STORAGE
2	(2)	BITSTRING	1	INITFLG0	FLAG BYTE 0
		1...		INITBLW	1 => LOC=BELOW SPECIFIED
		.1..		INITANY	1 => LOC=ANY SPECIFIED
		..1.		INITPRAL	VERIFYX INTERNAL PROPAGATION
		...1		INITVFYX	RACINIT VERIFYX INDICATOR

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		.... 1...		INITSYSN	1 - PARAMETER SPECIFIED THAT IS NOT COMPATIBLE WITH SYSTEM=YES
		.... .1..		INITNLOG	1 - LOG=NONE SPECIFIED
		11		*	RESERVED
3	(3)	BITSTRING	1	INITFLG1	FLAG BYTE 1
		11..		INITENVR	ENVIR - 00 CREATE, 01 CHANGE, 10 DELETE, 11 VERIFY
		..1.		INITNSMC	1 => NO STEP MUST COMPLETE
		...1		INITSUBS	SUBPOOL VALUE SPECIFIED
		.... 1...		INITPCHK	1 => NO PASSWORD PROCESSING TO BE PERFORMED
		.... .1..		INITNSTA	1 => STAT=NO SPECIFIED
		.... ..1.		INITULOG	1 => LOG=ALL SPECIFIED
		1		INITENCR	1 => ENCRYPT=NO SPECIFIED
4	(4)	ADDRESS	4	INITUPTR	ADDR OF USERID BUFFER
8	(8)	ADDRESS	4	INITPPTR	ADDR OF PASSWORD BUFFER
12	(C)	ADDRESS	4	INITSPTR	ADDR OF START PROC NAME
16	(10)	ADDRESS	4	INITIPTR	ADDR OF INSTALLATION INFO
20	(14)	ADDRESS	4	INITGPTR	ADDR OF GROUP NAME BUFFER
24	(18)	ADDRESS	4	INITNPTR	ADDR OF NEW PASSWORD BUFFER
28	(1C)	CHARACTER		INITEND1	END PART1

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
28	(1C)	STRUCTURE	20	INITPRM2	VER 1 REL 2
28	(1C)	ADDRESS	4	INITPGRP	ADDR OF PROGRAMMER NAME BUFFER

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
32	(20)	ADDRESS	4	INITACCP	ADDR OF ACCOUNT NUMBER BUFFER
36	(24)	ADDRESS	4	INITOIDP	ADDR OF MAGNETIC STRIPE CARD BUFFER
40	(28)	ADDRESS	4	INITTRMP	ADDR OF TERMINAL ID BUFFER
44	(2C)	ADDRESS	4	INITJOBP	ADDR OF JOB NAME
48	(30)	CHARACTER		INITEND2	END PART2

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
20	(14)	STRUCTURE	8	INITPRM3	VER 1 REL 3
20	(14)	ADDRESS	4	INITAPPP	ADDR APPLICATION NAME
24	(18)	ADDRESS	4	INITACEP	ADDR ACEE ANCHOR
28	(1C)	CHARACTER		INITEND3	END PART3

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
8	(8)	STRUCTURE	44	INITPRM4	RELEASE 1.9
8	(8)	UNSIGNED	1	INITSESN	SESSION TYPE - SEE TOKEN MAP FOR SPECIFIC VALUES
9	(9)	BITSTRING	1	INITFLG2	WORK UNIT IDENTITY FLAGS
		`1... ....`		INITRS	PART OF TRUSTED COMP BASE
		`.1.. ....`		INITRMT	THIS JOB FROM REMOTE NODE
		`..1. ....`		INITRSSP	TRUSTED KEYWORD SPECIFIED
		`...1 ....`		INITRMSP	REMOTE KEYWORD SPECIFIED
		`.... 1111`		*	RESERVED
10	(A)	SIGNED	2	*	RESERVED
12	(C)	ADDRESS	4	INITSLBP	SECLABL ADDRESS
16	(10)	ADDRESS	4	INITXNDP	EXENODE ADDRESS
20	(14)	ADDRESS	4	INITSIDP	SUSERID ADDRESS

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
24	(18)	ADDRESS	4	INITSNDP	SNODE ADDRESS
28	(1C)	ADDRESS	4	INITSGPP	SGROUP ADDRESS
32	(20)	ADDRESS	4	INITPOEP	POE ADDRESS
36	(24)	ADDRESS	4	INITUTKP	INPUT TOKEN ADDRESS
40	(28)	ADDRESS	4	INITSTKP	STOKEN ADDRESS
44	(2C)	ADDRESS	4	INITLSRP	LOGSTR ADDRESS
48	(30)	ADDRESS	4	INITOTKP	OUTPUT TOKEN ADDRESS
52	(34)	CHARACTER		INITEND4	END OF 1.9 PLIST

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
52	(34)	STRUCTURE	8	INITPRM5	RELEASE 1.9.2
52	(34)	ADDRESS	4	INITENVI	ENVRIN ADDRESS
56	(38)	ADDRESS	4	INITENVO	ENVROUT ADDRESS
60	(3C)	CHARACTER		INITEND5	END OF 1.9.2 PLIST

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	9	INITUSR	USERID BUFFER
0	(0)	ADDRESS	1	INITUSRL	USERID LENGTH
1	(1)	CHARACTER	8	INITUSRI	USERID

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	9	INITPAS	PASSWORD BUFFER
0	(0)	ADDRESS	1	INITPASL	PASSWORD LENGTH
1	(1)	CHARACTER	8	INITPASS	PASSWORD

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	9	INITGRP	GROUP NAME BUFFER
0	(0)	ADDRESS	1	INITGRPL	GROUP NAME LENGTH

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
1	(1)	CHARACTER	8	INITGRPN	GROUP NAME

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	9	INITNPA	NEW PASSWORD BUFFER
0	(0)	ADDRESS	1	INITNPAL	NEW PASSWORD LENGTH
1	(1)	CHARACTER	8	INITNPAS	NEW PASSWORD

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	256	INITOIDB	OID BUFFER
0	(0)	ADDRESS	1	INITOIDL	OID LENGTH
1	(1)	CHARACTER	255	INITOID	OID VALUE

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	9	INITENOD	EXECUTION NODE KEYWORD
0	(0)	UNSIGNED	1	INITENLN	LENGTH OF EXEC NODE DATA
1	(1)	CHARACTER	8	INITENNM	NAME OF EXECUTION NODE

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	9	INITSUID	SUBMITTERS USERID KEYWD
0	(0)	UNSIGNED	1	INITSILN	LENGTH OF SUBMIT USERID
1	(1)	CHARACTER	8	INITSINM	NAME OF SUBMITTER'S ID

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	9	INITSNOD	SUBMITTER'S NODE KEYWORD
0	(0)	UNSIGNED	1	INITSNLN	SUBMIT NODE DATA LENGTH

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
1	(1)	CHARACTER	8	INITSNNM	NAME OF SUBMITTER'S NODE

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	9	INITSGRP	SUBMITTER'S GROUP KEYWD
0	(0)	UNSIGNED	1	INITSGLN	SUBMIT GROUP DATA LENGTH
1	(1)	CHARACTER	8	INITSGNM	NAME OF SUBMIT GROUP

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	256	INITLGST	LOG STRING KEYWORD MAP
0	(0)	UNSIGNED	1	INITLSLN	LENGTH OF LOG STRNG DATA
1	(1)	CHARACTER	255	INITLGSD	LOG STRING DATA

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	14	INITENVD	ENVR OBJECT DATA STRUCTURE
0	(0)	UNSIGNED	4	INITELEN	ENVR OBJECT LENGTH
4	(4)	UNSIGNED	4	INITESLN	ENVR OBJECT STORAGE AREA LENGTH
8	(8)	ADDRESS	4	INITESAD	ENVR OBJECT STORAGE AREA ADDRESS
12	(C)	UNSIGNED	1	INITESSP	ENVR OBJECT STORAGE AREA SUBPOOL
13	(D)	UNSIGNED	1	INITESKY	ENVR OBJECT STORAGE AREA KEY

Cross Reference

Name	Hex Offset	Hex Value	Level
INITACCP	20		2
INITACEP	18		2

Name	Hex Offset	Hex Value	Level
INITANY	2	40	3
INITAPPP	14		2
INITBLW	2	80	3
INITELEN	0		2
INITENCR	3	01	3
INITEND1	1C		2
INITEND2	30		2
INITEND3	1C		2
INITEND4	34		2
INITEND5	34		2
INITENLN	0		2
INITENNM	1		2
INITENOD	0		1
INITENVD	0		1
INITENVI	2C		2
INITENVO	30		2
INITENVR	3	80	3
INITESAD	8		2
INITESKY	D		2
INITESLN	4		2
INITESSP	C		2
INITFLG0	2		2
INITFLG1	3		2
INITFLG2	9		2
INITGPTR	14		2
INITGRP	0		1
INITGRPL	0		2
INITGRPN	1		2
INITIPTR	10		2
INITJOBP	2C		2
INITLEN	0		2
INITLGSD	1		2
INITLGST	0		1
INITLSLN	0		2
INITLSRP	2C		2
INITNLOG	2	04	3

Name	Hex Offset	Hex Value	Level
INITNPA	0		1
INITNPAL	0		2
INITNPAS	1		2
INITNPTR	18		2
INITNSMC	3	20	3
INITNSTA	3	04	3
INITOID	1		2
INITOIDB	0		1
INITOIDL	0		2
INITOIDP	24		2
INITOTKP	30		2
INITPARM	0		1
INITPAS	0		1
INITPASL	0		2
INITPASS	1		2
INITPCHK	3	08	3
INITPGRP	1C		2
INITPOEP	20		2
INITPPTR	8		2
INITPRAL	2	20	3
INITPRM2	1C		1
INITPRM3	14		1
INITPRM4	8		1
INITPRM5	2C		1
INITRMSP	9	10	3
INITRMT	9	40	3
INITRS	9	80	3
INITRSSP	9	20	3
INITSESN	8		2
INITSGLN	0		2
INITSGNM	1		2
INITSGPP	1C		2
INITSGRP	0		1
INITSIDP	14		2
INITSILN	0		2
INITSINM	1		2

Name	Hex Offset	Hex Value	Level
INITSLBP	C		2
INITSNDP	18		2
INITSNLN	0		2
INITSNNM	1		2
INITSNOD	0		1
INITSPTR	C		2
INITSTKP	28		2
INITSUB#	1		2
INITSUBS	3	10	3
INITSUID	0		1
INITSYSN	2	08	3
INITTRMP	28		2
INITULOG	3	02	3
INITUPTR	4		2
INITUSR	0		1
INITUSRI	1		2
INITUSRL	0		2
INITUTKP	24		2
INITVFYX	2	10	3
INITXNDP	10		2

RLST

Common Name:	Request-specific portion of the RACROUTE REQUEST=LIST parameter list
Macro ID:	None
DSECT Name:	None
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Variable
Created by:	RACROUTE REQUEST=LIST macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=LIST routine.

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	28	RLSTPARM	LIST parameters
0	(0)	CHARACTER	2	RLSTSPNS	Subpool specifications
0	(0)	UNSIGNED	1	RLSTPSPN	Profile subpool number
1	(1)	UNSIGNED	1	RLSTNSPN	Tree node subpool number
2	(2)	UNSIGNED	1	RLSTCODE	Always set to 2
3	(3)	BITSTRING	1	RLSGFLAG	Flags:
		11..		RLSTOPT	Type of request: '00'B for create, '10'B for delete
		.1..		RLSTOWN	1 = add OWNER to access list with ALTER authority
		...1		RLSTLOC	1 = LOC=ABOVE specified
		.... 1...		RLSTREL	1 = RELEASE=1.8 specified
		.... .1..		RLSTR19	1 = RELEASE=1.9 specified
		.... ..1.		RLSTR192	1 = RELEASE=1.9.2 specified
		1		*	Reserved
1	(1)	CHARACTER	3	*	Reserved
4	(4)	ADDRESS	4	RLSTLIST	Address of resource name list
8	(8)	ADDRESS	4	RLSTACEE	Address of ACEE to use
12	(C)	ADDRESS	4	RLSTINST	Address of installation exit data field
16	(10)	ADDRESS	4	RLSTAPPL	Address of application name
20	(14)	ADDRESS	4	RLSTCLAS	Address of class name
24	(18)	ADDRESS	4	RLSTFLTP	Address of filter string

Cross Reference

Name	Hex Offset	Hex Value	Level
RLSTFLAG	3		2
RLSTACEE	8		2
RLSTAPPL	10		2
RLSTCLAS	14		2
RLSTCODE	2		2
RLSTFLTP	18		2
RLSTINST	C		2
RLSTLIST	4		2
RLSTLOC	3	10	3

Name	Hex Offset	Hex Value	Level
RLSTNSPN	1		3
RLSTOPT	3	80	3
RLSTOWN	3	20	3
RLSTPARM	0		1
RLSTSPN	0		3
RLSTREL	3	08	3
RLSTR19	3	04	3
RLSTR192	3	02	3
RLSTSPNS	0		2

RRPF

Common Name:	Resident Profile Map
Macro ID:	ICHRRPF
DSECT Name:	RRPF,DSPVOLS,DSPACCES,DSPINSTD, DSPDPTD,DSP2ACCS
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Subpool and Key:	Subpool 231 and key 0 when CSA profile requested; Subpool 229 and key 0 when private profile requested On VM, these subpools only apply within the RACFVM service machine.
Size:	1st Section: 136 bytes. 2nd Section: 2 bytes plus an unknown number of 6-byte fields at offset 2. 3rd Section: 2 bytes plus an unknown number of 9-byte fields at offset 2. 4th Section: 2 bytes plus a variable of unknown length at offset 2. 5th Section: 2 bytes plus an unknown number of 2-byte fields at offset 2. 6th Section: 35 bytes plus a variable of unknown length at offset 35. 7th Section: 2 bytes plus a variable of unknown length at offset 2. 8th Section: 2 bytes plus a variable of unknown length at offset 2.
Created by:	RACROUTE REQUEST=AUTH processing when CSA or private option is specified
Pointed to by:	ACEEAMP field of the ACEE data area, or returned in Register 1 after RACROUTE REQUEST=AUTH request
Serialization:	None
Function:	This area maps a profile for general resource used for authorization checking.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	136	RRPF	RESIDENT PROFILE MAP
0	(0)	UNSIGNED	4	DSPCORE	

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	UNSIGNED	1	RRPSP	AREA SUBPOOL NUMBER
1	(1)	ADDRESS	3	RRPLEN	TOTAL AREA LENGTH
4	(4)	CHARACTER	132	RRPVDATA	PROFILE DATA
4	(4)	CHARACTER	132	DSPSUB	
4	(4)	CHARACTER	44	DSPDSNM	RESOURCE NAME This name is also located in new structure below. This mapping maintained for compatibility for earlier releases
48	(30)	BITSTRING	1	DSPUACC	UNIVERSAL ACCESS
49	(31)	BITSTRING	1	DSPAUDIT	AUDIT FLAGS
50	(32)	BITSTRING	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)	ADDRESS	1	DSPLEVEL	RESOURCE LEVEL
52	(34)	SIGNED	4	DSPVOLOF	OFFSET TO VOLSER LIST
56	(38)	SIGNED	4	DSPACCOF	OFFSET TO ACCESS LIST
60	(3C)	CHARACTER	8	DSPCLASS	RESOURCE CLASS
68	(44)	BITSTRING	1	DSPGAUD	GLOBAL AUDIT FLAG
69	(45)	UNSIGNED	1	DSPVRSN	VERSION = 1
70	(46)	BITSTRING	1	DSPWARN	WARNING FLAG BIT 7 = 1 - RESOURCE HAS WARNING ATTRIBUTE
71	(47)	BITSTRING	1	DSPEOS	ERASE-ON-SCRATCH FLAG BIT 0 = 1 - DATASET WILL BE ERASED WHEN SCRATCHED
72	(48)	SIGNED	4	DSPINST	OFFSET TO INSTALLATION DATA
76	(4C)	ADDRESS	4	DSPNEXTP	ADDR NEXT MODEL
80	(50)	BITSTRING	1	DSPFNF	MODEL FOUND INDICATOR 0,FD -1,NFD
81	(51)	UNSIGNED	1	DSPSLVL	RESOURCE SECURITY LEVEL
82	(52)	SIGNED	2	DSPRTPD	RETENTION PERIOD
84	(54)	CHARACTER	8	DSPOWNER	RESOURCE OWNER

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
92	(5C)	CHARACTER	8	DSPNOTFY	USERID TO NOTIFY WHEN THIS PROFILE DENIES ACCESS
100	(64)	SIGNED	4	DSPDPTOF	OFFSET TO CATEGORY LIST
104	(68)	SIGNED	4	DSPPGMOF	OFFSET TO CONDITIONAL ACCESS LIST
108	(6C)	BITSTRING	1	DSPRESF	RESOURCE FLAG (ONLY FOR TAPE VOLUMES - BIT 0 = 1 VOLUME MAY ONLY CONTAIN ONE DATA SET - BIT 1 = 1 VOLUME CAN CONTAIN A TVTOC)
109	(6D)	BITSTRING	1	DSPTDAYS	DAYS THAT THE TERMINAL MAY NOT BE USED (BIT 0 - SUNDAY, BIT 1 - MONDAY,...
110	(6E)	CHARACTER	3	DSPLOGNT	EARLIEST TIME THAT THE TERMINAL BE USED.(HHMM)
113	(71)	CHARACTER	3	DSPLOGFT	LATEST TIME THAT THE TERMINAL BE USED.(HHMM)
116	(74)	CHARACTER	3	DSPTZONE	TIME OFFSET OF TERMINAL FROM THE CPU. (+ = EAST, - = WEST)
119	(77)	CHARACTER	1	*	RESERVED
120	(78)	CHARACTER	8	DSPSLABL	SECLABEL
128	(80)	CHARACTER	4	DSPDSNBF	Character form of offset to resource
128	(80)	SIGNED	4	DSPDSNOF	Offset to resource name in extended format
132	(84)	CHARACTER	4	DSPAPOFF	Offset to the application data.
132	(84)	SIGNED	4	DSPAPPOF	Offset to the application data.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSPVOLS	VOLSER LIST
0	(0)	UNSIGNED	2	DSPVOLCT	NUMBER OF ENTRIES
2	(2)	CHARACTER	6	DSPVOLSR (*)	VOLSERS

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSPACCES	ACCESS LIST
0	(0)	UNSIGNED	2	DSPACT	NUMBER OF ENTRIES
2	(2)	CHARACTER	9	DSPACCLE (*)	ACCESS LIST ENTRIES
2	(2)	CHARACTER	8	DSPAUSER	USERID/GRPNAME
10	(A)	BITSTRING	1	DSPACS	ACCESS AUTHORITY

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSPINSTD	INSTALLATION DATA
0	(0)	SIGNED	2	DSPLINST	LENGTH OF INSTALLATION DATA
2	(2)	CHARACTER	*	DSPIDATA	INSTALLATION DATA

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSPDPTD	CATEGORY LIST
0	(0)	SIGNED	2	DSPDPTCT	NUMBER OF CATEGORIES
2	(2)	SIGNED	2	DSPDEPT (*)	CATEGORY LIST

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSP2ACCS	Second Access List
0	(0)	UNSIGNED	2	DSP2GCT	Entry count
2	(2)	UNSIGNED	2	DSP2GLN	Access List Length
4	(4)	CHARACTER	20	DSP2ACCL	Entry structure
4	(4)	CHARACTER	8	DSP2ENT	Program Name / Flags
4	(4)	CHARACTER	1	DSPPGFLG	Flag byte
5	(5)	CHARACTER	7	DSPA2RST	The rest of name or flags
12	(C)	CHARACTER	8	DSP2USR	User/Group Id
20	(14)	BITSTRING	1	DSP2ACS	Access authority
21	(15)	UNSIGNED	2	DSP2GACS	Access Count
23	(17)	UNSIGNED	1	DSP2GVRL	Variable entity length
24	(18)	CHARACTER	*	DSP2GVAR	Variable entity info

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
24	(18)	CHARACTER	8	DSP2CLID	Class ID
32	(20)	CHARACTER	2	DSP2RSVD	Reserved
34	(22)	UNSIGNED	1	DSP2VENL	Variable Length
35	(23)	CHARACTER	*	DSP2VENT	Variable Entity

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSPBUF	Resource name in extended format
0	(0)	CHARACTER	2	DSPDLEN	Character form of resource name length
0	(0)	SIGNED	2	DSPDSNML	Resource name length
2	(2)	CHARACTER	*	DSPDSNME	Resource name

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	DSPAPPL	Structure of the application data.
0	(0)	SIGNED	2	DSPAPPLN	Length of the application data.
2	(2)	CHARACTER	*	DSPAPLDT	Application data.

Constants

Description of constants.

Len	Type	Value	Name	Description
1	DECIMAL	0	DSPA2DAT	Conditional data is present.
1	DECIMAL	0	DSPVR00	Version 0 profile present.
1	DECIMAL	1	DSPVR01	Version 1 profile present.
1	DECIMAL	1	DSPCURV	Version 1 profile is current version.

Cross Reference

Name	Hex Offset	Hex Value	Level
DSPACCES	0		1

Name	Hex Offset	Hex Value	Level
DSPACCLE	2		2
DSPACCOF	38		4
DSPACS	A		3
DSPACT	0		2
DSPAPLDT	2		2
DSPAPOFF	84		4
DSPAPPL	0		1
DSPAPPLN	0		2
DSPAPPOF	84		5
DSPAUDIT	31		4
DSPAUSER	2		3
DSPA2RST	5		4
DSPBUF	0		1
DSPCLASS	3C		4
DSPCORE	0		2
DSPDEPT	2		2
DSPDLEN	0		2
DSPDPTCT	0		2
DSPDPTD	0		1
DSPDPTOF	64		4
DSPDSNBF	80		4
DSPDSNM	4		4
DSPDSNME	2		2
DSPDSNML	0		3
DSPDSNOF	80		5
DSPEOS	47		4
DSPFNF	50		4
DSPGAUD	44		4
DSPIDATA	2		2
DSPINST	48		4
DSPINSTD	0		1
DSPLEVEL	33		4
DSPLINST	0		2
DSPLOGFT	71		4
DSPLOGNT	6E		4
DSPMDL	32	40	5

Name	Hex Offset	Hex Value	Level
DSPNEXTP	4C		4
DSPNOTFY	5C		4
DSPOWNER	54		4
DSPPGFLG	4		4
DSPPGMOF	68		4
DSPRESF	6C		4
DSPRTPD	52		4
DSPSLABL	78		4
DSPSLVL	51		4
DSPSUB	4		3
DSPTAPE	32	20	5
DSPTDAYS	6D		4
DSPTP	32	80	5
DSPTYPE	32		4
DSPTZONE	74		4
DSPUACC	30		4
DSPVOLCT	0		2
DSPVOLOF	34		4
DSPVOLS	0		1
DSPVOLSR	2		2
DSPVRSN	45		4
DSPWARN	46		4
DSP2ACCL	4		2
DSP2ACCS	0		1
DSP2ACS	14		3
DSP2CLID	18		3
DSP2ENT	4		3
DSP2GACS	15		3
DSP2GCT	0		2
DSP2GLN	2		2
DSP2GVAR	18		2
DSP2GVRL	17		3
DSP2RSVD	20		3
DSP2USR	C		3
DSP2ENVL	22		3
DSP2VENT	23		3

Name	Hex Offset	Hex Value	Level
RRPF	0		1
RRPLEN	1		3
RRPSP	0		3
RRPVDATA	4		2

RUTKN

Common Name:	User / Resource Security Token
Macro ID:	ICHRUTKN
DSECT Name:	TOKEN
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool, key, and residency are determined by user
Size:	80 bytes
Created by:	RACROUTE REQUEST=VERIFY, VERIFYX, or TOKENBLD
Pointed to by:	ACEETOKP and also returned as an output parameter from RACROUTE REQUEST=VERIFYX or TOKENBLD.
Serialization:	None
Function:	This mapping macro maps the RACF user security token and the RACF resource security token.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	TOKEN	UTOKEN/RTOKEN mapping element
0	(0)	UNSIGNED	1	TOKLEN	UTOKEN / RTOKEN length
1	(1)	UNSIGNED	1	TOKVERS	UTOKEN / RTOKEN version #
2	(2)	BITSTRING	1	TOKFLG1	miscellaneous flags
		`1... ....`		TOKENCR	Set to '1'B if the token is in an internal format (masked) and set to '0'B (external format) if it is not. TOKENCR must be set correctly regardless of the format of the token (internal or external).
		`.1.. ....`		*	reserved
		`..1. ....`		TOKLT19	Token created by pre-1.9 RACF call

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		...1		TOKVXPRP	VERIFYX propagation flag
		.... 1...		TOKUNUSR	NJE unknown user
		.... .1..		TOKLOGU	log user indicator
		.... ..1.		TOKRSPEC	RACF special indicator
		1		*	reserved
3	(3)	UNSIGNED	1	TOKSTYP	session type, defined below
4	(4)	BITSTRING	1	TOKFLG2	miscellaneous flags
		1...		TOKDFLT	default token
		.1..		TOKUDUS	undefined user
		..1.		*	reserved
		...1		TOKERR	token in error
		.... 1...		TOKTRST	part of trusted comp base
		.... .1..		TOKSUS	surrogate user ID
		.... ..1.		TOKREMOT	remote job indicator
		1		TOKPRIV	privileged indicator
5	(5)	UNSIGNED	1	TOKPOEX	port of entry class index
6	(6)	CHARACTER	2		reserved for expansion
8	(8)	CHARACTER	8	TOKSCL	SECLABL
16	(10)	CHARACTER	8	TOKXNOD	execution node
24	(18)	CHARACTER	8	TOKSUSR	submitting user IDD
32	(20)	CHARACTER	8	TOKSNOD	submitter node
40	(28)	CHARACTER	8	TOKSGRP	submitting group ID
48	(30)	CHARACTER	8	TOKPOE	port of entry (console ID, terminal ID)
56	(38)	CHARACTER	8	*	reserved for expansion
64	(40)	CHARACTER	8	TOKUSER	user ID
72	(48)	CHARACTER	8	TOKGRUP	group ID

Constants

Len	Type	Value	Name	Description

	TOKSTYP SESSION TYPE DEFINITIONS			

1	DECIMAL	1	TOKSAS	SYSTEM ADDRESS SPACE
1	DECIMAL	2	TOKCMND	COMMAND
1	DECIMAL	3	TOKCONS	CONSOLE OPERATOR
1	DECIMAL	4	TOKSTP	STARTED PROCEDURE
1	DECIMAL	5	TOKMNT	MOUNT
1	DECIMAL	6	TOKTSO	TSO LOGON
1	DECIMAL	7	TOKBCH	INTERNAL READER BATCH JOB
1	DECIMAL	8	TOKXBM	INTERNAL READER EXECUTION BATCH MONITOR
1	DECIMAL	9	TOKRJE	RJE OPERATOR
1	DECIMAL	10	TOKNJE	NJE OPERATOR
1	DECIMAL	11	TOKNJEUS	VERIFYX UNKNOWN USER ID TOKEN
1	DECIMAL	12	TOKEBCH	EXTERNAL READER BATCH JOB
1	DECIMAL	13	TOKRBCH	RJE BATCH JOB
1	DECIMAL	14	TOKNBCH	NJE BATCH JOB
1	DECIMAL	15	TOKNSYS	NJE SYSOUT
1	DECIMAL	16	TOKEXBM	EXTERNAL XBM
1	DECIMAL	17	TOKRXBM	RJE XBM
1	DECIMAL	18	TOKNXBM	NJE XBM
1	DECIMAL	19	TOKAPPC	APPC SESSION
1	DECIMAL	19	TOKLSESS	LAST CURRENTLY DEFINED SESSION

	TOKPOEX CLASS INDEX DEFINITIONS			
	SEE TOKCLTBL ARRAY BELOW			

Len	Type	Value	Name	Description
1	DECIMAL	1	TOKTERM	TERMINAL CLASS
1	DECIMAL	2	TOKCON	CONSOLE CLASS
1	DECIMAL	3	TOKJESI	JESINPUT CLASS
1	DECIMAL	4	TOKAPORT	APPCPORT CLASS
1	DECIMAL	4	TOKPLAST	LAST CLASS DEF

TOKVERS VERSION LEVEL DEFINITIONS

1	DECIMAL	1	TOKVER01	VERSION 1 TOKEN
1	DECIMAL	1	TOKCVER	LAST CURRENTLY DEFINED VERSION

Cross Reference

Name	Hex Offset	Hex Value	Level
TOKDFLT	4	80	3
TOKEN	0		1
TOKENCR	2	80	3
TOKERR	4	10	3
TOKGLG1	2		2
TOKFLG2	4		2
TOKGRUP	48		2
TOKLEN	0		2
TOKLOGU	2	04	3
TOKLT19	2	20	3
TOKPOE	30		2
TOKPOEX	5		2
TOKPRIV	4	01	3
TOKREMOT	4	02	3

Name	Hex Offset	Hex Value	Level
TOKRSPEC	2	02	3
TOKSCL	8		2
TOKSGRP	28		2
TOKSNOD	20		2
TOKSTYP	3		2
TOKSUS	4	04	3
TOKSUSR	18		2
TOKTRST	4	08	3
TOKUDUS	4	40	3
TOKKUNUSR	2	08	3
TOKUSER	40		2
TOKVERS	1		2
TOKVXPRP	2	10	3
TOKXNOD	10		2

RXTL

Common Name:	Request-specific portion of the RACROUTE REQUEST=EXTRACT parameter list
Macro ID:	None
DSECT Name:	None
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Variable
Created by:	RACROUTE REQUEST=EXTRACT macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=EXTRACT routine.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	EXTLIST	EXTRACT parameter list
0	(0)	SIGNED	2	EXTLEN	Parameter list length
2	(2)	BITSTRING	1	EXTFUNCT	Function code = x'82'

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
3	(3)	UNSIGNED	1	EXTTYPE	Request type: 1=Extract, 2=Encrypt 3=Extractn 4=Replace
4	(4)	UNSIGNED	1	EXTVER	Version number 0 or 1
5	(5)	BITSTRING	1	EXTFLAGS	Flag byte
		`1... ....`		EXTBRNCH	Branch entry requested
		`.1.. ....`		EXTENX	0 => Entity is specified and 1 => Entityx is specified
		`..11 1111`		*	Reserved
6	(6)	SIGNED	2	EXTOFF	Offset to variable part of list
8	(8)	ADDRESS	4	EXTENT	Addr of ENTITY
8	(8)	ADDRESS	4	EXTENTX	Addr of ENTITYX
12	(C)	CHARACTER		EXTEND	End of fixed part of parm

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
12	(C)	STRUCTURE	12	EXTEXT	TYPE=Extract parm list for release 1.6 and 1.7
12	(C)	ADDRESS	4	EXTCLAS	Addr of CLASS
16	(10)	SIGNED	4	EXTSP	Subpool for workarea
20	(14)	ADDRESS	4	EXTFLDS	Addr of data to be extracted. Data prefixed by 4-byte length
24	(18)	CHARACTER		EXTEND1	End of fixed part of parm

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
12	(C)	STRUCTURE	20	EXTENB	
12	(C)	ADDRESS	4	EXTSEGM	Addr of SEGMENT
16	(10)	ADDRESS	4	EXTSEGDT	Addr of SEGDATA
20	(14)	ADDRESS	4	EXTACEE	Addr of ACEE
24	(18)	ADDRESS	4	EXTVOL	Addr of VOLSER
28	(1C)	BITSTRING	4	EXTSPR	Special processing flags
28	(1C)	BITSTRING	3	EXTRES1	Reserved

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.... 1...		EXTMATCH	Match entity to generic
		.... .1..		EXTGEN	GENERIC flag
		.... ..1.		EXTDRV	DFP flag
		1		EXTFLAC	FLDACC flag
32	(20)	CHARACTER		EXTENDX	End of fixed part of parm

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
12	(C)	STRUCTURE	8	EXTENC	TYPE=ENCRYPT parameter list
12	(C)	ADDRESS	4	EXTDATA	Addr of data to be encrypted. Data prefixed by 1-byte length
16	(10)	SIGNED	4	EXTMETH	Encryption method: 1 = RACF DES method, 2 = RACF hashing method, 3 = installation defined method, 4 = NBS DES method

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	FIELDS	Map of FIELDS parameter
0	(0)	SIGNED	4	FLDCNT	Number of field names
4	(4)	CHARACTER	8	FLDNAME (*)	Individual field names

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	SEGDATS	Map segdata parameter
0	(0)	SIGNED	4	SEGFLEN	Size of data
4	(4)	CHARACTER	*	SEGFDTA	Segment data

Constants

Len	Type	Value	Name	Description
2	DECIMAL	24	EXTEXTL	Length of release 1.6 or 1.7 parameters
2	DECIMAL	44	EXTRL	
2	DECIMAL	20	EXTENCL	Length of encrypt parameters

Cross Reference

Name	Hex Offset	Hex Value	Level
EXTACEE	14		2
EXTBRNCH	5	80	3
EXTCLAS	C		2
EXTDATA	C		2
EXTDRV	1F	02	3
EXTENB	C		1
EXTENC	C		1
EXTEND	C		2
EXTENDX	20		2
EXTEND1	18		2
EXTENT	8		2
EXTENTX	8		3
EXTENX	5	40	3
EXTEXT	C		1
EXTFLAC	1F	01	3
EXTFLAGS	5		2
EXTFLDS	14		2
EXTFUNCT	2		2
EXTGEN	1F	04	3
EXTLEN	0		2
EXTLIST	0		1
EXTMATCH	1F	08	3
EXTMETH	10		2
EXTOFF	6		2
EXTRES1	1C		3
EXTSEGDT	10		2
EXTSEGM	C		2
EXTSP	10		2

Name	Hex Offset	Hex Value	Level
EXTSPR	1C		2
EXTTYPE	3		2
EXTVER	4		2
EXTVOL	18		2
FIELDS	0		1
FLDCNT	0		2
FLDNAME	4		2
SEGDATS	0		1
SEGFDTA	4		2
SEGFLEN	0		2

RXTW

Common Name:	RACROUTE REQUEST=EXTRACT result area mapping
Macro ID:	IRRPRXTW
DSECT Name:	EXTWKEA, EXTWANM, EXTWABG, EXTWADP, EXTWARM, EXTWAS1, EXTWAS2, EXTWAS3, EXTWAS4, EXTWAAC
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: 229 or subpool supplied by issuer of RACROUTE REQUEST=EXTRACT Key: 0 or as determined by the subpool of the issuer of RACROUTE REQUEST=EXTRACT
Size:	1st Section: 72 bytes. 2nd Section: 8 bytes for Release 1.7; variable for Release 1.8 and subsequent releases. 3rd through 11th section: Work attributes data–4-byte field followed by variable data.
Created by:	RACROUTE REQUEST=EXTRACT
Pointed to by:	Register 1 after RACROUTE REQUEST=EXTRACT has been issued
Serialization:	None
Function:	This area maps the fixed portion of the results from RACROUTE REQUEST=EXTRACT and the work attributes data that is extracted from the ACEE.

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	EXTWKEA	
0	(0)	SIGNED	4	EXTWPLEN	
0	(0)	UNSIGNED	1	EXTWSP	Area subpool
1	(1)	UNSIGNED	3	EXTWLN	Area length

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
4	(4)	SIGNED	2	EXTWOFF	Offset from EXTWKEA to start of optional returned fields
6	(6)	CHARACTER	1	EXTFLAG	Flag Byte
		1...		EXTGENRC	Generic profile retrieved for TYPE=EXTRACTN
		.111 1111		*	Reserved bits
7	(7)	UNSIGNED	1	EXTWVERS	Version of Result Area
8	(8)	SIGNED	2	EXTWAOFF	Offset from EXTWKEA to start of optional Work Attributes area when extracting from the ACEE
10	(A)	CHARACTER	6	*	Reserved
16	(10)	CHARACTER	3	EXTWPRLN	User's or default primary language
19	(13)	CHARACTER	3	EXTWSELN	User's or default secondary language
22	(16)	CHARACTER	1	EXTWPRUS	Whether the reported primary language is defined in the user profile (U) or is the installation default (S)
23	(17)	CHARACTER	1	EXTWSEUS	Whether the reported secondary language is defined in the user profile (U) or is the installation default (S)
24	(18)	CHARACTER	8	EXTWUID	Specified or current user's id
32	(20)	CHARACTER	8	EXTWGRP	Specified user's default group or current user's current connect group
40	(28)	CHARACTER	32	*	Reserved
72	(48)	CHARACTER		EXTWEND	End of fixed part

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
72	(48)	STRUCTURE	8	EXTWOPT	Optional fields for TYPE=EXTRACT or EXTRACTN and Release 1.8 or later
72	(48)	CHARACTER	8	EXTWPSWD	Encoded password for TYPE=EXTRACT and Release 1.7 or earlier

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWANM	WORKATTR - User name
0	(0)	SIGNED	4	EXTWNMLN	Length of user name
4	(4)	CHARACTER	*	EXTWNAME	User name for SYSOUT

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWABG	WORKATTR - Building name
0	(0)	SIGNED	4	EXTWBDLN	Length of building name
4	(4)	CHARACTER	*	EXTWBLDG	Building name for delivery

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWADP	WORKATTR - Department name
0	(0)	SIGNED	4	EXTWDTLN	Length of department name
4	(4)	CHARACTER	*	EXTWDEPT	Department name for delivery

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWARM	WORKATTR - Room name
0	(0)	SIGNED	4	EXTWRMLN	Length of room name
4	(4)	CHARACTER	*	EXTWROOM	Room for delivery

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWAS1	WORKATTR - SYSOUT line 1
0	(0)	SIGNED	4	EXTWS1LN	Length of SYSOUT line 1
4	(4)	CHARACTER	*	EXTWSYS1	SYSOUT delivery line 1

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWAS2	WORKATTR - SYSOUT line 2
0	(0)	SIGNED	4	EXTWS2LN	Length of SYSOUT line 2
4	(4)	CHARACTER	*	EXTWSYS2	SYSOUT delivery line 2

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWAS3	WORKATTR - SYSOUT line 3
0	(0)	SIGNED	4	EXTWS3LN	Length of SYSOUT line 3
4	(4)	CHARACTER	*	EXTWSYS3	SYSOUT delivery line 3

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWAS4	WORKATTR - SYSOUT line 4
0	(0)	SIGNED	4	EXTWS4LN	Length of SYSOUT line 4
4	(4)	CHARACTER	*	EXTWSYS4	SYSOUT delivery line 4

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	EXTWAAC	WORKATTR - Account number
0	(0)	SIGNED	4	EXTWATLN	Length of account number
4	(4)	CHARACTER	*	EXTWACCT	Account number

Cross Reference

Name	Hex Offset	Hex Value	Level
EXTFLAG	6		2
EXTGENRC	6	80	3
EXTWAAC	0		1
EXTWABG	0		1
EXTWACCT	4		2
EXTWADP	0		1
EXTWANM	0		1
EXTWAOFF	8		2

Name	Hex Offset	Hex Value	Level
EXTWARM	0		1
EXTWAS1	0		1
EXTWAS2	0		1
EXTWAS3	0		1
EXTWAS4	0		1
EXTWATLN	0		2
EXTWBDLN	0		2
EXTWBLDG	4		2
EXTWDEPT	4		2
EXTWDTLN	0		2
EXTWEND	48		2
EXTWGRP	20		2
EXTWKEA	0		1
EXTWLN	1		3
EXTWNAME	4		2
EXTWNMLN	0		2
EXTWOFF	4		2
EXTWOPT	48		1
EXTWPLEN	0		2
EXTWPRLN	10		2
EXTWPRUS	16		2
EXTWPSWD	48		2
EXTWRMLN	0		2
EXTWROOM	4		2
EXTWSELN	13		2
EXTWSEUS	17		2
EXTWSP	0		3
EXTWSYS1	4		2
EXTWSYS2	4		2
EXTWSYS3	4		2
EXTWSYS4	4		2
EXTWS1LN	0		2
EXTWS2LN	0		2
EXTWS3LN	0		2
EXTWS4LN	0		2
EXTWUID	18		2

Name	Hex Offset	Hex Value	Level
EXTWVERS	7		2

SAFP

Common Name:	System Authorization Facility (SAF) Router Parameter List
Macro ID:	ICHSAFP
DSECT Name:	SAFP
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	104 bytes
Created by:	RACROUTE macro
Pointed to by:	RACROUTE MF=E or MF=S places the address into R1 before invoking SAF (MVS) or the external security product (VM).
Serialization:	None
Function:	This macro is the descriptor for data passed to the SAF router (MVS) or the external security product (VM) by the RACROUTE macro.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	104	SAFP	
0	(0)	SIGNED	4	SAFPRRET	RACF or installation exit Return Code
4	(4)	SIGNED	4	SAFPRREA	RACF or installation exit Reason Code
8	(8)	SIGNED	2	SAFPPLN	Length of SAFP parameter list (in bytes)
10	(A)	UNSIGNED	1	SAFPVRRL	RACF Version/Release list indicator (values defined below)
11	(B)	CHARACTER	1	*	Reserved
12	(C)	SIGNED	2	SAFPREQT	Request number (values defined below)
14	(E)	BITSTRING	1	SAFPFLGS	Flags
		`1... ....`		SAFPMSGR	1 - Message return requested
		`.1.. ....`		SAFPR18	1 - Release 1.8 or higher function was requested

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		..1.		SAFPSUPP	1 - Message suppression on
		...1		SAFPDCPL	1 - DECOUPL=yes
		.... 1...		SAFPSYST	1 - SYSTEM=yes
		.... .111		*	RESERVED
15	(F)	UNSIGNED	1	SAFPMSPL	Subpool to be used for messages to be returned, if SAFPMSGR is on
16	(10)	ADDRESS	4	SAFPREQR	Requestor name address (points to an 8-byte character field)
20	(14)	ADDRESS	4	SAFPSUBS	Subsystem name address (points to an 8-byte character field)
24	(18)	ADDRESS	4	SAFPWA	SAF work area address
28	(1C)	ADDRESS	4	SAFPMSAD	Upon return, will contain the address of the message(s) returned from the invoked function (if SAFPMSGR is on), or zero if no message is returned
32	(20)	ADDRESS	4	*	RESERVED
36	(24)	SIGNED	4	SAFPRACP	Offset to RACF related parameter list from base address of SAFP
40	(28)	SIGNED	4	SAFPSFRC	SAF Return Code
44	(2C)	SIGNED	4	SAFPSFRS	SAF Reason Code
48	(30)	SIGNED	2	SAFPPLNX	Length of SAFP extension parameter list (in bytes)
50	(32)	SIGNED	2	SAFPOLEN	Length of Original Plist
52	(34)	ADDRESS	4	SAFPRETD	Address of returned data
56	(38)	ADDRESS	4	SAFPFLAT	Address of flat parameter
60	(3C)	ADDRESS	4	SAFPECB1	Address of ECB1
64	(40)	ADDRESS	4	SAFPECB2	Address of ECB1
68	(44)	ADDRESS	4	SAFPPREV	Address of previous flat list
72	(48)	ADDRESS	4	SAFPNEXT	Address of next flat list
76	(4C)	ADDRESS	4	SAFPORIG	Address of original list
80	(50)	SIGNED	4	SAFPFLEN	Flat parameter list length

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
84	(54)	SIGNED	4	SAFPUSRW	User Word - Identifier
88	(58)	ADDRESS	4	SAFPPREE	Address of Pre-Processing exit
92	(5C)	ADDRESS	4	SAFPPOST	Address of Post-Processing exit
96	(60)	ADDRESS	4	SAFPSYNC	Address of Synchronous ECB
100	(64)	UNSIGNED	1	SAFPSKEY	Requestors Storage Key
101	(65)	UNSIGNED	1	SAFPMODE	Requestors Addressing mode
102	(66)	UNSIGNED	1	SAFPSBYT	Status Byte
		1...		SAFPGCS	1 - Request came from GCS
		.1..		SAFPSFSU	1 - SFS user accessing own file or directory (used for SFSAUTOACCESS processing)
103	(67)	UNSIGNED	1	*	Reserved
104	(68)	CHARACTER		*	

Constants

Len	Type	Value	Name	Description
4	DECIMAL	104	SAFPLEN	
CONSTANTS FOR REQUEST NUMBER VALUES				
1	DECIMAL	1	SAFPAU	RACHECK - Authorization function
1	DECIMAL	2	SAFPFAU	FRACHECK - Fast authorization function
1	DECIMAL	3	SAFPLIS	RACLIST - In-storage list building function
1	DECIMAL	4	SAFPDEF	RACDEF - Definition function
1	DECIMAL	5	SAFPVER	RACINIT - Verification function
1	DECIMAL	6	SAFPEXT	RACXTRT - Extract function
1	DECIMAL	7	SAFPDIR	RACDAUTH - Directed Authorization function
1	DECIMAL	8	SAFPTKMP	RACTKSRV - Security Token services
1	DECIMAL	9	SAFPVERX	RACINIT - envir=verifyx
1	DECIMAL	10	SAFPTKXT	RACTKSRV - extract Token services
1	DECIMAL	11	SAFPTBLD	RACINIT - Token build services
1	DECIMAL	12	SAFPEXTB	RACXTRT - Branch Entry

Len	Type	Value	Name	Description
1	DECIMAL	13	SAFPAUD	RACAUDIT - Audit Service
1	DECIMAL	14	SAFPSTAT	RACSTAT - Status Service
	Constants for SAFPRRET and SAFPRREA for the requests handled in SAF Constants for TOKENBLD request			
1	DECIMAL	8	SAFPTBRC	SAFPTBLD request SAF r.c
	The following reason codes are used :			
1	DECIMAL	0	SAFPTBUT	TOKNOUT missing - 9C7 SAF abend
1	DECIMAL	4	SAFPTBUL	TOKNOUT length too small: on return the length field in the token has the correct length - 9C7 SAF abend
1	DECIMAL	8	SAFPTBTK	Invalid token data - 9C7 SAF abend
1	DECIMAL	12	SAFPTBSL	STOKEN length too small: on return the length field in the token has the correct length - 9C7 SAF abend
1	DECIMAL	16	SAFPTBUB	TOKNOUT length too large: on return the length field in the token has the correct length
1	DECIMAL	20	SAFPTBSB	STOKEN length too large: on return the length field in the token has the correct length
1	DECIMAL	24	SAFPTBV0	A token passed in did not have its version set - 9C7 SAF abend
1	DECIMAL	28	SAFPTBIL	TOKNIN length too small: on return the length field in the token has the correct length - 9C7 SAF abend
1	DECIMAL	28	SAFPTBIB	TOKNIN length too large: on return the length field in the token has the correct length
	Constants for VERIFYX request			
1	DECIMAL	60	SAFPVXRC	SAFPVERX request SAF r.c
	The following reason codes are used :			
1	DECIMAL	0	SAFPVXNR	RACF not available
1	DECIMAL	4	SAFPVXOP	Old Password required
1	DECIMAL	8	SAFPVXUS	Userid required
1	DECIMAL	12	SAFPVXEF	ESTAE failed

Len	Type	Value	Name	Description
1	DECIMAL	16	SAFPVXUT	TOKNOUT keyword missing - 9C7 SAF abend
1	DECIMAL	20	SAFPVXUL	TOKNOUT length too small: on return the length field in the token has the right length - 9C7 SAF abend
1	DECIMAL	24	SAFPVXTK	Invalid token data - 9C7 SAF abend
1	DECIMAL	28	SAFPVXSL	STOKEN length too small: on return the length field in the token has the right length - 9C7 SAF abend
1	DECIMAL	32	SAFPVXUB	TOKNOUT length too large: on return the length field in the token has the right length
1	DECIMAL	36	SAFPVXSB	STOKEN length too large: on return the length field in the token has the right length
1	DECIMAL	40	SAFPVXV0	A token passed in did not have its version set - 9C7 SAF abend
1	DECIMAL	44	SAFPVXIL	TOKNIN length too small: on return the length field in the token has the correct length - 9C7 SAF abend
1	DECIMAL	48	SAFPVXIB	TOKNIN length too large: on return the length field in the token has the correct length
Constants for VERIFY request				
1	DECIMAL	64	SAFPVYRC	SAFPVER request SAF r.c
The following reason codes are used :				
1	DECIMAL	0	SAFPVYTK	Invalid token data - 9C7 SAF abend
1	DECIMAL	4	SAFPVYUL	TOKNIN length too small: on return the length field in the token has the right length - 9C7 SAF abend
1	DECIMAL	8	SAFPVYSL	STOKEN length too small: on return the length field in the token has the right length - 9C7 SAF abend
1	DECIMAL	12	SAFPVYUB	TOKNIN length too large: on return the length field in the token has the right length
1	DECIMAL	16	SAFPVYSB	STOKEN length too large: on return the length field in the token has the right length

Len	Type	Value	Name	Description
1	DECIMAL	20	SAFPVYV0	A token passed in did not have its version set - 9C7 SAF abend
Constants for Version/Release				
1	DECIMAL	16	SAFPCURR	Current level of RACF
1	DECIMAL	0	SAFPRL19	Indicates RACF 1.9.0
1	DECIMAL	2	SAFPR192	Indicates RACF 1.9.2
1	DECIMAL	16	SAFPR530	Indicates RACF 530

Cross Reference

Name	Hex Offset	Hex Value	Level
SAFP	0		1
SAFPDCPL	E	10	3
SAFPECB1	3C		2
SAFPECB2	40		2
SAFPFLAT	38		2
SAFPFLEN	50		2
SAFPFLGS	E		2
SAFPGCS	66	80	3
SAFPMODE	65		2
SAFPMSAD	1C		2
SAFPMSGR	E	80	3
SAFPMSPL	F		2
SAFPNEXT	48		2
SAFPOLEN	32		2
SAFPORIG	4C		2
SAFPPLN	8		2
SAFPPLNX	30		2
SAFPPOST	5C		2
SAFPPREE	58		2
SAFPPREV	44		2
SAFPRACP	24		2
SAFPREQR	10		2
SAFPREQT	C		2
SAFPRETD	34		2
SAFPRREA	4		2
SAFPRRET	0		2

Name	Hex Offset	Hex Value	Level
SAFPR18	E	40	3
SAFPSBYT	66		2
SAFPSFRC	28		2
SAFPSFRS	2C		2
SAFPSFSU	66	40	3
SAFPSKEY	64		2
SAFPSUBS	14		2
SAFPSUPP	E	20	3
SAFPSYNC	60		2
SAFPSYST	E	08	3
SAFPUSRW	54		2
SAFPVRRL	A		2
SAFPWA	18		2

SAFV

Common Name:	SAF Router Vector Table Map
Macro ID:	ICHSAFV
DSECT Name:	SAFV
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	SAFV
	Offset 0
	Length 4
Storage Attributes:	Subpool: 245 Key: 0 Residency: below 16M
Size:	32 bytes
Created by:	External Security Manager Initialization (z/VM)
Pointed to by:	CVTSAF
Serialization:	None
Function:	Provides PL/S and BAL mapping of the SAF router vector table.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	32	SAFV	SAF VECTOR TABLE
0	(0)	CHARACTER	4	SAFVIDEN	IDENTIFYING LITERAL FOR DUMPS 'SAFV'
4	(4)	UNSIGNED	1	SAFVVRSN	TABLE VERSION NUMBER - '00'X

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
5	(5)	CHARACTER	3	*	RESERVED
8	(8)	ADDRESS	4	*	RESERVED
12	(C)	ADDRESS	4	SAFVSAFR	ADDRESS OF THE RACROUTE support code
16	(10)	ADDRESS	4	*	RESERVED
20	(14)	ADDRESS	4	*	RESERVED
24	(18)	ADDRESS	4	*	RESERVED
28	(1C)	ADDRESS	4	SAFVRAC2	ADDRESS OF Callable Services Router
32	(20)	CHARACTER		*	ENSURE DOUBLE WORD LENGTH

Constants

Len	Type	Value	Name	Description
4	DECIMAL	32	SAFVLEN	LENGTH OF THE SAF ROUTER VECTOR TABLE
4	CHARACTER	SAFV	SAFVIDC	LITERAL VALUE TO BE STORED IN SAFVIDEN
1	DECIMAL	0	SAFVVNC	LITERAL VALUE OF SAF VERSION NUMBER STORED IN SAFVVRSN

Cross Reference

Name	Hex Offset	Hex Value	Level
SAFV	0		1
SAFVEXIT	8		2
SAFVEXUS	18		2
SAFVIDEN	0		2
SAFVRACR	10		2
SAFVRACT	14		2
SAFVSAFR	C		2
SAFVCRSN	4		2

STAT

Common Name:	Request-specific portion of the RACROUTE REQUEST=STAT parameter list
Macro ID:	None

DSECT Name:	None
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: Determined by caller Key: Determined by caller Residency: Determined by caller
Size:	Variable
Created by:	RACROUTE REQUEST=STAT macro
Pointed to by:	Address of SAFP plus offset in SAFPRACP
Serialization:	None
Function:	Maps the request-specific portion of the parameter list passed to RACROUTE REQUEST=STAT routine.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	STATPARM	STAT parameters
0	(0)	ADDRESS	4	STATCLAS	Address of class entry
4	(4)	ADDRESS	4	STATCDTP	Output: address of class entry in the CDT
8	(8)	SIGNED	4	*	Present only if RACROUTE is used
8	(8)	UNSIGNED	2	STATLEN	Length of this parameter list
10	(A)	CHARACTER	2	*	Reserved

Cross Reference

Name	Hex Offset	Hex Value	Level
STATCDTP	4		2
STATCLAS	0		2
STATLENL	8		3
STATPARM	0		1

TSRV

Common Name:	Request-specific portion of the RACROUTE REQUEST=TOKENMAP/ TOKENXTR parameter list (request section)
Macro ID:	None
DSECT Name:	None
Owning Component:	Resource Access Control Facility (SC1BN)
Eye-Catcher ID:	None

Storage Attributes:

Subpool
Determined by caller

Key
Determined by caller

Residency
Determined by caller

Size: Varies

Created by: RACROUTE REQUEST=TOKENMAP/TOKENXTR macro

Pointed to by: Address of SAFP plus offset in SAFPRACP

Serialization: None

Function: Maps the request-specific portion of the parameter list passed to the RACROUTE REQUEST=TOKENMAP/TOKENXTR routine

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	24	TSRVPARM	TOKENMAP/TOKENXTR parameters
0	(0)	ADDRESS	4	TSRVTKIN	TOKNIN pointer
4	(4)	ADDRESS	4	TSRVACEE	ACEE pointer
8	(8)	ADDRESS	4	TSRVTKOT	TOKNOUT pointer
12	(C)	BITSTRING	1	TSRVFLGS	Flag byte
		`1... ....`		TSRVFMT	Format of output token for TOKENMAP: 1 = encrypt, 0 = decrypt
		`.111 1111`		*	Reserved
13	(D)	CHARACTER	1	*	Reserved
14	(E)	UNSIGNED	2	TSRVLEN	Length of this parameter list
16	(10)	CHARACTER	8	*	Reserved

Cross Reference

Name	Hex Offset	Hex Value	Level
TSRVACEE	4		2
TSRVFLGS	C		2
TSRVFMT	C	80	3
TSRVLEN	E		2
TSRVPARM	0		1
TSRVTKIN	0		2
TSRVTKOT	8		2

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

© Copyright IBM Corp. 1990, 2022

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This publication primarily documents intended programming interfaces that allow the customer to write programs to obtain services of an external security manager.

This document also contains information that is NOT intended to be used as programming interfaces. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

NOT programming interface information

End of NOT programming interface information

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

© Copyright IBM Corp. 1990, 2022

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/support/pages/node/6019492), SA22-7935
- Open Systems Adapter-Express Integrated Console Controller User's Guide (https://www.ibm.com/support/pages/node/6019810), SC27-9003
- Open Systems Adapter-Express Integrated Console Controller 3215 Support (https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SA23-2247
- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SC14-7580

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330

- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350033/$file/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350152/$file/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350151/$file/ifc1000_v2r5.pdf), GC35-0151

Related Products

z/OS

- *Common Programming Interface Communications Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf)*, SC26-4399
- z/OS and z/VM: Hardware Configuration Definition Messages (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sc342668/$file/cbdm100_v2r5.pdf), SC34-2668
- z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sc342670/$file/eequ100_v2r5.pdf), SC34-2670
- z/OS: Network Job Entry (NJE) Formats and Protocols (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa320988/$file/hasa600_v2r5.pdf), SA32-0988
- z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa760169/$file/glpa300_v2r5.pdf), SA76-0169
- z/OS: Language Environment Concepts Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380687/$file/ceea800_v2r5.pdf), SA38-0687
- z/OS: Language Environment Debugging Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5ga320908/$file/ceea100_v2r5.pdf), GA32-0908
- z/OS: Language Environment Programming Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380682/$file/ceea200_v2r5.pdf), SA38-0682
- z/OS: Language Environment Programming Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380683/$file/ceea300_v2r5.pdf), SA38-0683
- z/OS: Language Environment Runtime Messages (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380686/$file/ceea900_v2r5.pdf), SA38-0686
- z/OS: Language Environment Writing Interlanguage Communication Applications (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380684/$file/ceea400_v2r5.pdf), SA38-0684
- z/OS: MVS Program Management Advanced Facilities (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa231392/$file/ieab200_v2r5.pdf), SA23-1392
- z/OS: MVS Program Management User's Guide and Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa231393/$file/ieab100_v2r5.pdf), SA23-1393

XL C++ for z/VM

- XL C/C++ for z/VM: Runtime Library Reference, SC09-7624
- XL C/C++ for z/VM: User's Guide, SC09-7625

Index

A

ACEE data area
  changing 255
  creating with RACROUTE REQUEST=VERIFY macro 162

C

coding macros 3
combination field definitions
  format of 295
combination field definitions (RACF database)
  definition 295
connect template
  contents of 315
continuation coding
  example 5
continuation lines 5
control from external security manager
  to RACROUTE support code
    through BALR 339

D

data areas
  TSRV 442
data set template
  contents of 317
database profiles, storage requirements 295

E

example
  of continuation coding 5
External Security Manager (ESM)
  non-RACF 337

F

FACILITY class
  activating 13
  creating the ICHCONN profile 12
fields
  character 295
  date 294
  integer 295
  time 295
FRACHECK macro
  chart of parameters by release 210
  description 208
  list form 211
  used to check a user's authorization to a resource 208

G

general resource
general resource *(continued)*
  fields in the profile 322
general template
  contents of 322
General-Use Mappings
  for RACROUTE 347
group
  template for database 298
group profile
  description of the fields 298
group template
  contents 298

I

ICHCONN profile in the FACILITY class
  creating 12
ICHRFX02 exit routine
  reason codes 124, 125, 211
Initializing modules
  RPIATGCS 337
  RPIUCMS 337

L

Limited function RACROUTE macro
  on z/VM 267
list
  parameter
    passed to RACROUTE REQUEST=TOKENMAP routine 442
    passed to RACROUTE REQUEST=TOKENXTR routine 442

M

macro forms
  what they mean 1
macros
  description (on z/VM prior to Release 1.9) 267
  list of 7
  RACDEF macro 214
  RACHECK macro 234
  RACINIT macro 249
  RACLIST Macro 261
  RACROUTE macro 8
  RACROUTE macro on z/OS 273
  RACROUTE macro on z/VM 267
  RACROUTE REQUEST=AUDIT 27
  RACROUTE REQUEST=AUTH macro 35
  RACROUTE REQUEST=DEFINE macro 55
  RACROUTE REQUEST=DIRAUTH 86
  RACROUTE REQUEST=EXTRACT macro 92
  RACROUTE REQUEST=FASTAUTH 121
  RACROUTE REQUEST=LIST Macro 128
  RACROUTE REQUEST=STAT 141, 143

macros *(continued)*
RACROUTE REQUEST=STAT macro 138
RACROUTE REQUEST=TOKENBLD macro 144
RACROUTE REQUEST=TOKENMAP macro 153
RACROUTE REQUEST=TOKENXTR 158
RACROUTE REQUEST=VERIFY macro 162
RACROUTE REQUEST=VERIFYX macro 185
RACSTAT 276
RACSTAT macro 273
RACXTRT macro 277
that invoke RACF 7

P

parameter list
passed to RACROUTE REQUEST=
TOKENMAP routine 442
TOKENXTR routine 442
postprocessing exit routines
FRACHECK macro
reason codes 211
RACROUTE REQUEST=FASTAUTH macro
reason codes 124, 125
profile
contents of a data set profile on MVS 317
contents of a general resource profile 322
contents of a group profile 298
contents of a user profile 301
profile (RACF database)
repeat groups 294
profile checking 50, 53, 247

R

RACDEF macro
chart of parameters by release 224
description 214
example 227, 228
execute form 231
list form 228
return codes 225
standard form 214
syntax 214, 231
used to define, modify, or delete resource profiles 214
RACF
system macros 7, 207
RACF database
general template 322
group template 298
reserved templates 335
user template 301
RACF database, restructured
connect template 315
RACF macros
coding 3
FRACHECK macro 208
list of 207
RACSYNC macro on z/VM
syntax 204
sample 4
that invoke RACF 207
RACF system macros
forms of the macro 1
RACHECK macro
chart of parameters by release 242
description 234
example 243
execute form 247
list form 245
profile checking 247
return codes 242
standard form 235
syntax 235, 245, 247
used to provide authorization checking 234
RACINIT macro
chart of parameters by release 254
description 249
example 133, 134, 256, 257, 264, 265
execute form 259
list form 257
return codes 255
standard form 249
syntax 249
used to provide user identification and verification 249
RACLIST macro
chart of parameters by release 263
description 261
example 264
execute form 266
list form 265
return codes 263
standard form 261
syntax 261
used to build in-storage profiles 261
RACROUTE
sample program for shared user IDs 345
RACROUTE Interface on z/VM
to non-RACF ESM 337
RACROUTE macro
chart of parameters by RACROUTE keywords 9
chart of RACROUTE keywords by parameters 9
description 8
description (on z/OS prior to Release 1.9) 273
examples 22
execute form 24
list form 23
modify form 26
return codes 20
return codes - z/VM only 21
standard form 16
syntax 16
used to invoke SAF 8
RACROUTE parameter list
ICHSAFP 341
RACROUTE REQUEST=
TOKENMAP
macro 442
routine 442
TOKENXTR
macro 442
routine 442
RACROUTE REQUEST=AUDIT macro
description 27
execute form 33
list form 32
modify form 34
return codes 30

RACROUTE REQUEST=AUDIT macro *(continued)*
standard form 28
syntax 28, 32–34
RACROUTE REQUEST=AUTH macro
description 35
execute form 50
list form 47
modify form 53
profile checking 50, 53
return codes 44
return codes (CDT-default) 47
standard form 35
syntax 35, 47, 50, 53
used to provide authorization checking 35
RACROUTE REQUEST=DEFINE macro
description 55
example 73, 74
execute form 78
list form 74
modify form 82
standard form 56
syntax 56, 78, 82
used to define, modify, or delete resource profiles 55
RACROUTE REQUEST=DIRAUTH macro
examples 89
execute form 90
list form 89
modify form 91
return codes 88
standard form 86
syntax 86, 89–91
RACROUTE REQUEST=DIRAUTH Macro
description 86
used to 86
RACROUTE REQUEST=EXTRACT macro
description 92
example 109, 110, 112
execute form 116
list form 113
modify form 118
return codes 106
standard form 93
syntax 93, 113, 116, 118
used to retrieve fields from user profile 92
RACROUTE REQUEST=FASTAUTH macro
description 121
execute form 126
list form 125
used to check a user's authorization to a resource 121
RACROUTE REQUEST=LIST macro
description 128
example 133
execute form 135
list form 134
modify form 137
return codes 132
standard form 128
syntax 128
used to build in-storage profiles 128
RACROUTE REQUEST=STAT
return codes 140
RACROUTE REQUEST=STAT macro
description 138
execute form 141
RACROUTE REQUEST=STAT macro *(continued)*
list form 141
modify form 143
used to determine if RACF is active 138
RACROUTE REQUEST=TOKENBLD macro
description 144
example 148
execute form 150
list form 148
modify form 152
return codes 147
standard form 144
syntax 144
used to build a UTOKEN 144
RACROUTE REQUEST=TOKENMAP macro
description 153
examples 155
execute form 156
list form 155
modify form 157
return codes 155
standard form 153
syntax 153, 155–157
used to access token fields 153
RACROUTE REQUEST=TOKENXTR macro
description 158
examples 160
execute form 161
list form 160
modify form 162
return codes 159
standard form 158
syntax 158, 160–162
used to extract a token 158
RACROUTE REQUEST=VERIFY macro
description 162
example 176
execute form 179
list form 176
modify form 182
return codes 173
standard form 163
syntax 163, 176, 179, 182
used to provide user identification and verification 162
RACROUTE REQUEST=VERIFYX macro
description 185
example 196
execute form 199
list form 196
modify form 201
return codes 193
standard form 185
syntax 185, 196, 199, 201
used to build a UTOKEN 185
RACROUTE requests
authorizing virtual machines to issue RACROUTE requests 12
controlling use by virtual machines on z/VM 12
RACSTAT macro
chart of parameters by release 274
description 273
execute form 276
list form 275
used to determine if RACF is active 273

RACSYNC macro on z/VM
  accesses returned parameter lists and loads registers 204
  description of 204
  syntax 204
RACXTRT macro
  chart of parameters by release 286
  description 277
  execute form 290
  list form 288
  return codes 286
  standard form 277
  syntax 277, 288, 290
  used to retrieve fields from user profile 277
reason codes
  from ICHRFX02 exit routine 124, 125, 211
repeat groups (RACF database) 294
requests
  passed to RACROUTE REQUEST=
    TOKENMAP routine 442
    TOKENXTR routine 442
resowner field, DFP 96
return codes
  from FRACHECK macro 211
  from RACDEF macro 225
  from RACHECK macro 242
  from RACINIT macro 255
  from RACLIST macro 263
  from RACROUTE macro 20
  from RACROUTE macro on z/VM 21, 270
  from RACROUTE REQUEST=AUTH macro 44
  from RACROUTE REQUEST=AUTH macro (CDT-related) 47
  from RACROUTE REQUEST=DEFINE macro 70
  from RACROUTE REQUEST=DIRAUTH macro 87
  from RACROUTE REQUEST=EXTRACT macro 106
  from RACROUTE REQUEST=FASTAUTH macro 124
  from RACROUTE REQUEST=LIST macro 131
  from RACROUTE REQUEST=STAT 140
  from RACROUTE REQUEST=TOKENBLD macro 147
  from RACROUTE REQUEST=TOKENMAP macro 155
  from RACROUTE REQUEST=TOKENXTR macro 159
  from RACROUTE REQUEST=VERIFYX macro 193
  from RACXTRT macro 286
  from SAF 20
  unique to z/VM 21
routines
  RACROUTE REQUEST=
    TOKENMAP 442
    TOKENXTR 442
RPIATGCS
  requirements for providing 338
RPIGCS LOADLIB
  requirements for providing 338
RPIUCMS
  requirements for providing 337

S

SAF return codes 20
sample
  RACROUTE program for shared user IDs 345
STOKEN values
  overridden by UTOKEN 144, 185
storage requirements, profiles 295
system macros
  what the forms mean 1

T

templates
  connect 315
  data set 317
  general 322
  group 298
  reserved 335
  user 301
templates (RACF database)
  combination field definitions 295
  format of field definitions 293
  repeat groups 294
TSRV data area 442

U

unit of work
  security information propagated from use session 162
user identification and verification
  using RACROUTE REQUEST=VERIFY macro 162
user profile
  description of the fields 301
user template
  contents of 301
UTOKEN
  building 144, 185

Product Number: 5741-A09

Printed in USA

SC24-6324-73